elektor

WINTER WIDGETS EDITION – 132 PAGES PACKED WITH NEW DESIGNS

LED Ring Light

the #1 macro enhancer

● Aviation Band Scanner | Temperature & RH Logger | 13-IN-A-BOX

The 7-uP Alarm Clock | **FPGA Programming** | Embedded Linux

Arduino On Course | Realistic LED Candle ● Open Source Hardware

● **1956 stroboscope** ● Frontline Breaking News ● Elektor World

Contents

● Tech the Future

● Magazine

•Community

Winter Widgets Edition #1

Welcome to the first double-size
edition of Elektor for the year 2013,
covering January and February. From
past experience and trials we've
come to understand that the Elektor
readership is really keen on having a
big bunch of circuits, ideas, reviews,
lab reports, and various unfinished
symphonies, to get them through
the *solstitio brumali* — the very
dead of winter. Our double editions
traditionally are the best sold and
draw the highest response rates,
which is gratifying to say the least.
Hence our decision to publish two of them every year.
On the following 100-odd editorial pages we aim to strike a balance
between small circuits and larger ones, including installments of Embedded
Linux, Arduino on Course, and Taming the Beast (called FPGA). The contents
also reflect the new 'dot xyz' structure of the magazine introduced in
December 2012 and in letters to our members — with some tweaks and
refinements thanks to 'critical acclaim' received from you.
This edition further underscores the importance and dynamic position of
that hot burning e-furnace called Elektor-Labs. The 13-IN-A-BOX project is
intentionally carried on these pages as a diamond in the rough just waiting
for your code, suggestions for the casing, improvements, and anything else
you think is worthwhile to engineer the project towards the 'prestigious
project' status. The Elektor lab folks and your fellow readers extend a cordial
welcome at www.elektor-labs.com.
Finally, I'm calling on all readers of Elektor USA to negotiate publication of
their article with me. That's right, **you** write. I'm convinced there's a lot of
talent, ingenuity and promise out there, if not in winter time then a bit later
for sure.

Jan Buiting, Managing Editor

The Team

Managing Editor:	Jan Buiting
Publisher:	Hugo Van haecke
Membership Manager:	Shannon Barraclough
International Editorial Staff:	Harry Baggen, Eduardo Corral, Wisse Hettinga, Denis Meyer, Jens Nickel
Laboratory Staff:	Thijs Beckers, Ton Giesberts, Luc Lemmens, Clemens Valens, Raymond Vermeulen, Jan Visser
Graphic Design & Prepress:	Giel Dols, Jeanine Opreij, Mart Schroijen
Online Manager:	Daniëlle Mertens
Managing Director:	Don Akkermans

Our network

Connects you to

265646 members
in 83 countries

Supporting Companies

Not a supporting company yet?

Contact Peter Wostrel (peter@smmarketing.us, Phone 978-281-7708, Fax 978-281-7706)
to reserve your own space for the next edition of our members' magazine

Elektor World

Compiled by
Wisse Hettinga

Every day, every hour, every minute, at every given moment designers and enthusiasts are thinking up, tweaking, reverse-engineering and developing new electronics. Chiefly for fun, but occasionally fun turns into serious business. Elektor World connects some of the events and activities — for fun and for business.

Hai (Helen) Li

🇺🇸 The female engineers archive

VERNON — Coworker CJ from Circuit Cellar magazine came up with an interesting subject — women. And particularly — women in engineering. CJ writes: "In the late 1980s and early 1990s, the engineers who worked with MCUs, ran engineering companies, and taught EE/ECE courses at world-renowned universities were mustachioed Caucasian males sporting starched button-down shirts, thick bifocals, and pocket protectors.

OK, perhaps I'm overgeneralizing *just a bit*. There were a few cool guys (maybe 10?) whose personalities (enthusiastic, creative, curious), interests (good music, literature, motor bikes), and styles (professional yet lab-casual) presaged the coming of the gregarious, unpretentious, multitalented 21st-century techies who are developing the world's game-changing technologies.

So, what's the problem? We now have amazing technologies to work with, right? Well, the most notable issue is that they were mostly all *guys*. For far too long, female engineers have been underrepresented in the tech community. But during the last few years things have been changing, and we're seeing many female engineers really break out from the pack!

In 2012, *Circuit Cellar* began spotlighting these talented innovators. Here are a few:

- Ayse Kivilcim Coskun ("A Vision for 3-D Stacked Systems," *Circuit Cellar* 264)
- Hai (Helen) Li ("Embedded Inquiries," *Circuit Cellar* 267)
- Jan Axelson ("Debugging USB Firmware," *Circuit Cellar* 268)

And there's more to come in 2013! Check out our "Female Engineers" archive: http://circuitcellar.com/category/female-engineers/".

🇮🇳 Meet your new colleague

PUNE — While in Europe business and colleges are struggling to get new engineering students, in India it is easy to capture more than 100 in one room. This is the Bharati Vidyapeeth's College of Engineering for Women in Pune. The girls are studying IT and Embedded Electronics Engineering and they are eager and enthusiastic. This school is one of the around 80 engineering colleges in Pune alone. With the current number of engineers arriving from colleges in Europe

there is a good chance that one of these young ladies is your new colleague shortly.

🇬🇧 'Underground' radio

LONDON — Remember Elektor's Summer Edition 2012 where we asked you to think of weird PCB designing — like changing the London 'Tube' Map into a working circuit? Yuri Suzuki, a proficient designer, is thinking weird along the same lines. He changed the Tube Map into a working radio, and the result is up for viewing in London's Design Museum. Also check his website — Yuri is into some very scintillating designs.

🇳🇱 The case against the killer robots

AMSTERDAM — It does not happen often: a calling for a public debate about the desirability of a technology that is still at least 20 or 30 years away from even

being possible. But that''s exactly what Human Rights Watch and Harvard Law School's International Human Rights Clinic are doing by publishing their report *Losing Humanity: The Case Against Killer Robots*. Follow the comments on Techthefuture.com

🇳🇱 Teachers! Leave them kids alone

LIMBRICHT — By invitation a group of teachers 'abandoned the kids' but just for a day. Elektor wanted to hear more about today's tech

education in The Netherlands and offered selected e-teachers a tour of the Labs. It turned out a pleasant, informal get-together. If you happen to be around you are always welcome — just drop

us a mail or call and we will arrange something.

🇳🇱 HP35 draws a calculated response

Dear Elektor Editors — I was delighted to read your "Retronics" article on the HP35 calculator in the November 2012 edition — nostalgia! I remember being one of the earliest users, and the HP35 at the time set me back 2,100 Dutch guilders.

I still own that particular HP35 and it still works. It's just the on/off switch that's a bit unreliable at times. I also own a 31E, 71B, 18C, 28C, and a 200LX. Can you suggest a good home for these calculators, like a museum? Right now they're doing little except gathering dust and I am not comfortable with just throwing them away.
Gerard Keijser, The Netherlands

Editor Jan Buiting comments:
Time and again I am gratified and delighted by my readers' response to "the old pages" towards the back of the magazine, called *Retronics*. Gerard

Community

kindly complied with my request to send the calculators to Elektor in return for a copy of my 'Retronics' book. The collection is pictured here for all Elektor USA readers to admire and reminisce about.

Chippies... It's (acid) bath time!

MOSCOW — *Meanwhile in Russia* a small company called ZeptoBars, every week has a fresh picture of a famous IC topology (like the 8080 micro). They dissolve the IC housings using concentrated sulfuric acid bath at 300 °C for half an hour. Afterwards, if necessary, a hot concentrated nitric acid bath removes the remaining debris of the package. Well, we wouldn't say it's a *lovely* bath... but definitely it's worth doing! It doesn't matter how many different shots of silicon chips we've seen, they're always amazing. We just can't get enough. High resolution pics of world famous designs are relatively easy to find, but other not so run-of-the-mill ICs definitely are not. We guess that most companies just try to keep them safe from immediate reverse engineering, at least during the *first wave*. But on the other hand, we wouldn't want to miss these electronic eye candies.
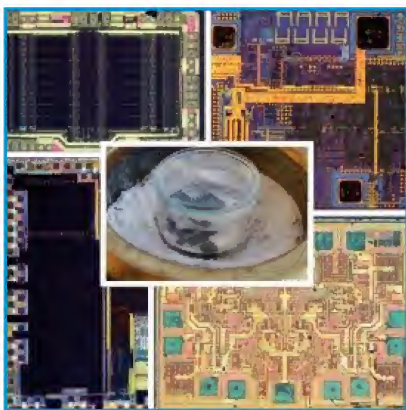
Ping-pong with Arduino

CHIASSO — If you go to the Swiss Arduino offices in Chiasso for a meeting, most likely you end up around the Ping-Pong table. Not to play this famous game, but quite simply they don't have a table large enough to host a party of six. Elektor was there to sign a World Wide distribution contract for Arduino products and yes, we had some food and drinks to celebrate this. From left to right you see Don trying to convince Gianlucca, Massimo trying to check his mail, David trying to
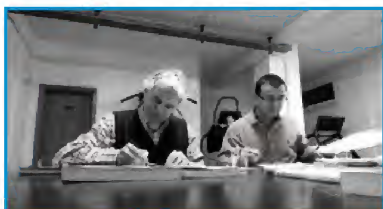
hide from the picture, Eduardo trying to show his best looks. Stay tuned for more on Arduino.

and now for something completely different...

Feedback! on Current Transformers

A nice technical thread developed following the *Current Transformers* article in the October 2012 edition. Managing Editor Jan Buiting was honored to be Cc'd into the correspondence. Watch how the feedback loop evolves, and the specialists gather.

Hello Jan,

Since I design CTs for the aerospace industry, I found the "Current Transformers" article by Ed Dinning quite interesting. I do have one comment on Figure 5. The author shows the RL burden resistor on The CT secondary side of the rectifier diodes. This will cause the meter reading to vary with the temperature-dependent forward drop of the diodes. Since the CT is a current source, putting the RL burden resistor on the meter side of the diodes will make the meter reading independent of the diode drops. I showed this method for connecting the burden resistor in my "Emergency Load Generator Meter" article in 02-2012 Elektor, Fig 2. I used Schottky diodes to reduce the VA requirement on the CT.
I also discussed the advantages of toroidal transformer cores in an article in the May 1996 issue of Popular Electronics, pg 53 entitled "All About Toroidal Transformers".
Best regards,
Chuck Hansen (USA)

Hi Chuck,
thanks for the mail; yes I totally agree with your comments ref position of the burden and the tempco of the diode forward drop.
The intention of this article was to provide a simple, isolated indicating instrument that the average hobbyist could use, admittedly with some non-linearity at the bottom end of the scale.
If Schottky diodes are used, typically 1 A types, the 50 mA current should have little effect on their V_f.
Accuracy could also have been improved if a lower flux density had been used, but the core chosen was of a type that is easily available, or even re-claimed, from old equipment.
As a matter of interest what flux density do you work at with your toroidals?
When I worked at a switch gear manufacturer many years ago we were using special alloys and operating at 0.01 T.
BR, Ed Dinning (UK)

Hello Ed,
I liked your idea of repurposing old transformers. It's hard to change the voltage ratio of some commercially available transformers by adjusting the secondary turns because the secondary winding is on the inside of the bobbin, and the primary is wound over top of it for best flux coupling. Adding extra turns on the outside of the primary is also difficult since the core window is usually filled to capacity to minimize transformer weight (cost). In aerospace, minimum weight is a priority, so I push the CT flux density higher than the utilities do. The requirements differ depending on the use for the CT, but generally with silicon iron (B_{max} 16 kgauss or 1.6 T), I limit the full-load flux density to 3200 gauss (0.32 T).
We yanks seem to have an aversion to the metric system for reasons that escape me. Generator designers still work flux density in lines/sq inch!
With differential protection (two CTs connected out-of-phase across a protection circuit burden) the CTs cannot cause a false DP trip at the worst-case through-fault current, which is the current deliverd to a fault downstream of the zone protected by the CTs. I limit the flux density at the worst-case fault current, with allowance for sub-transient current, to about 1.4 T at maximum core temperature.
With the specified secondary burden and rated primary current, the phase angle error must not exceed 20 minutes, and the phase angle errors of all the CTs in the differential protection zone must not differ by more than 5 minutes from the mean of their phase angle errors. The difference in secondary current magnitudes between any two CTs must not exceed 0.25% of the mean of their secondary current magnitudes.
If the CT is used for overcurrent protection, or current limiting with an alternator voltage regulator, the core area I use depends on the circuit burden resistance plus the number of rectifier diodes. For metering and instrumentation, the military/aerospace specs defines the 400 Hz burden impedance to be $1.48 + j2.09\ \Omega \pm 10\%$.
We mainly use 3% silicon iron toroidal cores for our CTs, but other alloys are available for special applications:
Satmu metal (a special grain-oriented 65% nickel-iron alloy that is annealed in a magnetic field, developed by Telcon Ltd in the UK; the original developers of Mu metal) Deltamax Round Orthonol; Supermalloy powder metal for frequency > 2 kHz.
BR, Chuck Hansen

Hi Chuck,
 thanks for the info. Magnetics was very badly taught in UK schools, first in inches etc, then in cgs. It's only when you get to uni and they teach in SI (not MKS) that it all becomes clear. There is a society here to abolish the centimeter (a good idea, as school have really screwed up metrication).

Most of the transformers here have the primary wound first, making them easier to modify; also a lot of them have pri and sec side by side for safety reasons (creepage and clearance; easy to re-wind but lousy regulation).
The only time we depart from pri inside is for audio transformers and those on switch mode power supplies, where pri and sec are sectionalized and interleaved for best coupling.
I've done quite a bit of aerospace work; one design used supermendur running at 2 T / 400 Hz and force cooled. You needed to wear hearing defenders due to magnetostriction. It was for the TRU for a military aircraft.
A further interesting design was for frequency convertors on a military transport aircraft. 3-phase to 18-phase, rated for frequency wild operation (380 to 620 Hz), cooled with JET-A1. It may have simplified the alternators and improved power factor but it certainly complicated the windings and the electronics!
I deliberately avoided anything to do with j operator and power factor for the Elektor article.
BR, Ed Dinning
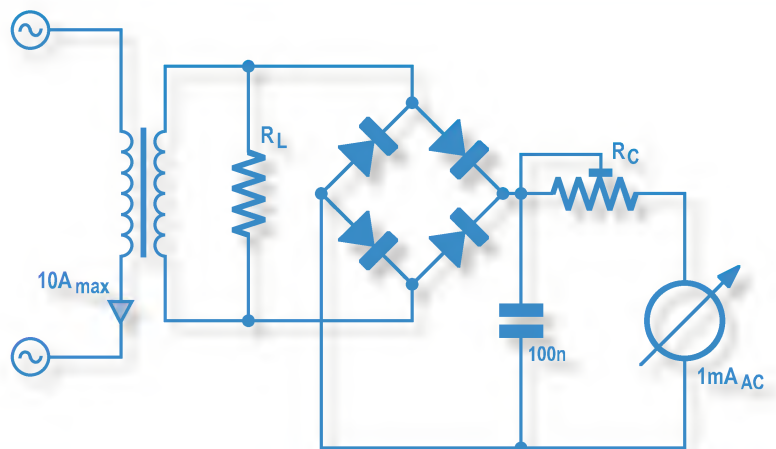
Hi Ed,
I agree with you about the "j" operator. When we began test metering CTs I received permission to use a 2R56 (2.56 ohm) wire wound resistor for the burden. The measurements all came out the same.
BR, Chuck Hansen

Hello,
I have an improvement regarding the example #2 of your "Design Tip: Current Transformers" article in the October 2012 issue. Example #2 shows the output of the current transformer connected to the load resistor (which must always be connected, as you properly stated). However the bridge rectifier takes the VOLTAGE signal from the load resistor and because of the forward voltage drop of the bridge, the output is somewhat non-linear to the point of having no response for small current signals.
A better solution is to rectify the output current of the transformer secondary FIRST, then load the rectified current at the output of the bridge with the load resistor. As the output of the current transformer is a current and acts as a current source, the same current must flow through the load resistor from the bridge rectifier (the transformer just needs to supply a little more voltage to overcome the diode V_f to allow the secondary current flow). Now the current through the load resistor is unidirectional and a simple RC filter before the meter will smooth the signal and scale it to the meter. The meter can respond to the smallest secondary currents without worry about the diode bridge voltage drop. You did mention the effect of the bridge rectifier voltage drop in the last paragraph, and I agree with the math and methodology in the rest of the article.
Thank you for a good lesson about current transformers; I hope this helps to enlighten engineers and hobbyists to their proper use.
KR, Jim Mettler, Triad Magnetics

RL78 Green Energy Challenge
Winners Announcement

The RL78 Green Energy Challenge set forth to revolutionize the way engineers approach new designs in a world where low-power consumption, high efficiency and renewable resources are integral parts of daily life. Participants from over 67 countries were invited to showcase their skills in developing green energy designs for applications — such as energy harvesting, metering, low power and control systems — using Renesas' RL78 microcontrollers and a robust software environment, powered by Renesas and its alliance partners.

Thank you for everyone's participation and congratulations to the winners!

The complete contest entries and abstracts are posted at

www.circuitcellar.com/contests/renesasRL78challenge/

First Prize
Electrostatic Cleaning Robot

Scott Potter (United States)

Solar tracking mirrors, called heliostats, are an integral part of Concentrating Solar Power (CSP) plants. They must be kept clean to help maximize the production of steam, which generates power. Using an RL78, the innovative Electrostatic Cleaning Robot provides a reliable cleaning solution that's powered entirely by photovoltaic cells. The robot traverses the surface of the mirror and uses a high voltage AC electric field to sweep away dust and debris.

Second Prize
Cloud Electrofusion Machine

Michael Hamilton (United States)

Using approximately 400 times less energy than commercial electrofusion machines, the Cloud Electrofusion Machine is designed for welding 0.5" to 2" polyethylene fittings. The RL78-controlled machine is designed to read a barcode on the fitting which determines fusion parameters and traceability. Along with the barcode data, the system logs GPS location to an SD card, if present, and transmits the data for each fusion to a cloud database for tracking purposes and quality control.

3

Third Prize
he Sun Chaser: A GPS Reference Station

Sjoerd Brandsma (Netherlands)

The Sun Chaser is a well-designed, solar-based energy harvesting system that automatically recalculates the direction of a solar panel to ensure it is always facing the sun. Mounted on a rotating disc, solar panel's orientation is calculated using the registered GPS position. With an external compass, the internal accelerometer, a DC motor and stepper motor, you can determine the solar panel's exact position. The system uses the Renesas RDKRL78G13 evaluation board running the Micrium µC/OS-III real-time kernel.
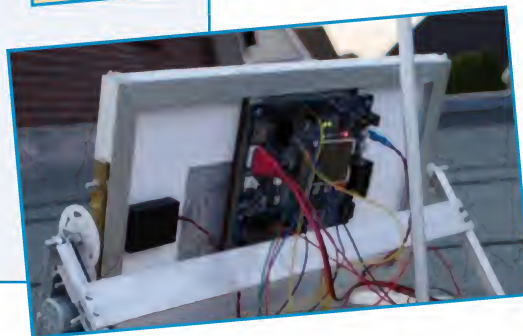
Honorable Mention
Water Heater by Solar Concentration

Pierre Berquin (France)

This solar water heater is powered by the RL78 evaluation board and designed to deflect concentrated amounts of sunlight onto a water pipe for continual heating. The deflector, armed with a counterweight for easy tilting, automatically adjusts the angle of reflection for maximum solar energy using the lowest power consumption possible.

Honorable Mention
Air Quality Mapper

Raul Alvarez Torrico (Bolivia)

Want to make sure the air along your daily walking path is clean? The Air Quality Mapper is a portable device designed to track levels of CO_2 and CO gasses for constructing "Smog Maps" to determine the healthiest routes. Constructed
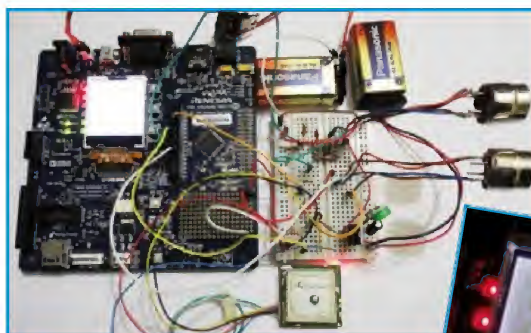
with an RDKRL78G13, the Mapper receives location data from its GPS module, takes readings of the CO_2 and CO concentrations along a specific route and stores the data in an SD card. Using a PC, you can parse the SD card data, plot it, and upload it automatically to an online MySQL database that presents the data in a Google map.

Honorable Mention
High-Altitude Low-Cost Experimental Glider (HALO)

Jens Altenburg (Germany)

The "HALO" experimental glider project consists of three main parts. A weather balloon is the

the way weather data is gathered and eases construction of local measurement networks while being optimized for low energy usage and long battery life.

carrier section. A glider (the payload of the balloon) is the return section. A ground base section is used for communication and display telemetry data (not part of the contest project). Using the REFLEX flight simulator for testing, the glider has its own micro-GPS receiver, sensors and low-power MCU unit. It can take off, climb to pre-programmed altitude and return to a given coordinate.

Honorable Mention
Portable Power Quality Meter
Andrè Barbosa (Brazil)
Monitoring electrical usage is becoming increasingly popular in modern homes. The

Portable Power Quality Meter uses an RL78 MCU to read power factor, total harmonic distortion, line frequency, voltage, and electrical consumption information and stores the data for analysis.

Honorable Mention
Wireless Remote Solar-Powered "Meteo Sensor"
Grzegorz Kaczmarek (Poland)
You can easily measure meteorological parameters with the "Meteo Sensor". The RL78 MCU-based design takes cyclical measurements of temperature, humidity, atmospheric pressure, and supply voltage, and shares them using digital radio transceivers. Receivers are configured for listening of incoming data on the same radio channel. It simplifies

The Most Popular 8-bit Microcontrollers

Easy Migration

Secure Supply Chain

Unified Development Environment

Lowest Power

MICROCHIP
PIC® MCU
MICROCHIP
MICROCHIP
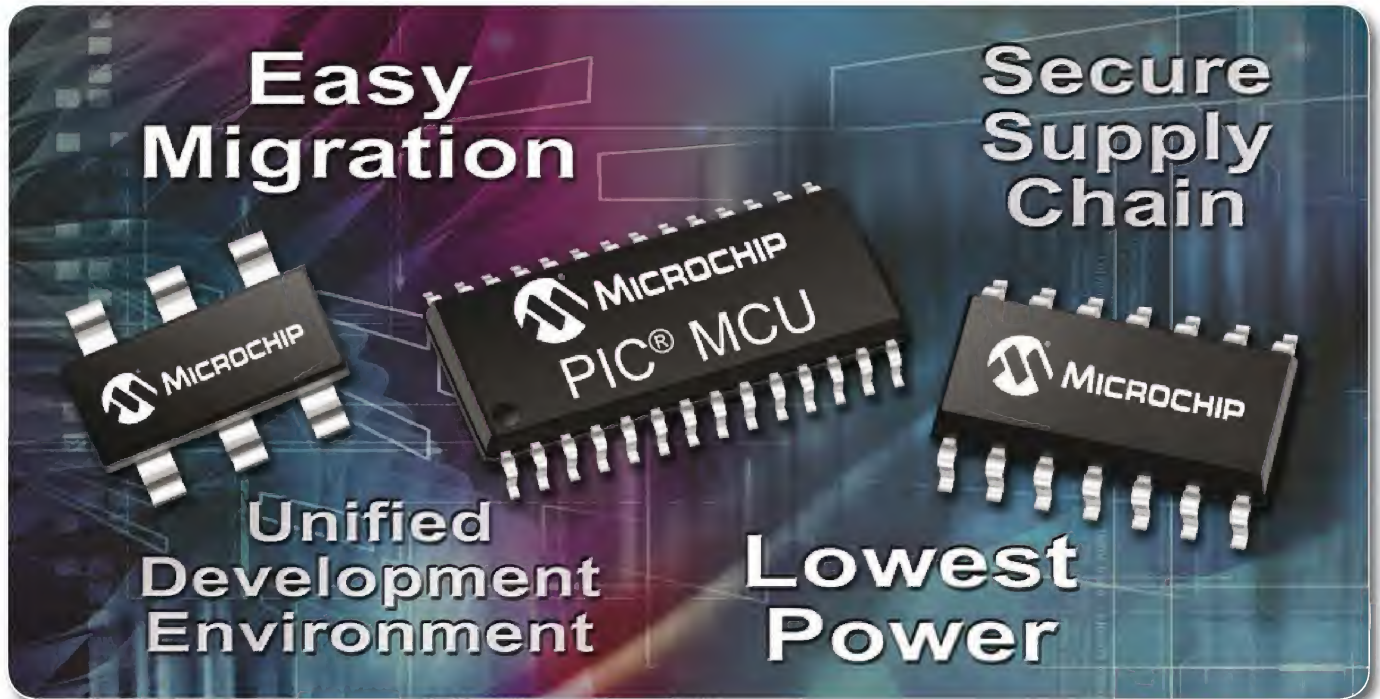
See what makes Microchip the most popular choice for embedded designers:

- Broad portfolio of more than 325 8-bit PIC® microcontrollers
- Easy migration with pin and code compatibility
- Industry's lowest active and sleep power consumption
- Integrated peripherals for USB, CAN and Ethernet with free software
- Interface to the world with LCD drivers and capacitive touch
- Continuous innovation with 70 new 8-bit MCUs in the last two years
- MPLAB® IDE is free, and supports ALL of Microchip's microcontrollers
- Low-cost development tools help speed up prototyping efforts
- Comprehensive technical documentation, app notes and code examples
- World-class 24/7 technical support and training

GET STARTED IN 3 EASY STEPS

1. Purchase the XLP 8-bit Development Board
2. Download free MPLAB® IDE
3. Order samples and start designing!
   www.microchip.com/8bit

XLP 8-bit Development Board
(DM240313)

MICROCHIP

Microcontrollers • Digital Signal Controllers • Analog • Memory • Wireless

Frontline breaking news

By Clemens Valens
(Elektor .Labs)

Elektor dot Labs is at the heart of Elektor, it is the place where all electronics related designs start. Projects and ideas are posted on the website; circuits get developed, debugged, tried and tested in our labs, and progress and results are reported back to you. Elektor dot Labs is also the frontline where all the action takes place. Here are some heartbeats from the front.

Up, up with the amps and volts

One of the first projects posted on .LABS was not a project, but a request for a laboratory power supply unit. Not a normal, plain vanilla lab PSU, no, the original poster (OP) wanted one that can supply up to 40 volts instead of the oh-so boring and common "30 volts max." lab PSU. Apparently the OP was not capable of designing such a PSU himself and so he (or she) came to us. And since one of our tasks is designing for you, we accepted the challenge and assigned one of our engineers on it.

Now I must admit that this project is a bit of a slow burner, and that it is still far from finished, but last week Ton Giesberts showed me his proof-of-concept and what do you guess? It worked! Not perfectly, but in a proof-of-concept kind of way, with some oscillations and hum here and there, you know how it goes. The thing is that Ton decided to try a new method of his own devising. Instead of using a variable-output voltage regulator he chose to use a fixed-output regulator and trick it into thinking that it is a variable-output regulator. Or something like that. You better ask Ton if you want to know the ins and outs of his design. BTW, these details happen to be on .LABS, on the page of this project, so check it out if PSUs get you excited.

One of the reasons that the project advanced so slowly is a capacitor that had to be ordered from the other side of the world. It is a very low-ESR part and the Chinese didn't have any left. When they finally rediscovered one that had dropped from the table and rolled under a filing cabinet, it took forever to put it into an envelope and ship it to Ton.

The OP had specified 2 ampères for the PSU, but Ton feels that's not enough. 3 amps would be better according to him. Or 6 amps. So I guess you have to be patient and wait a bit longer before this design will be finished and gets published in the magazine.

See how Ton is progressing:
www.elektor-labs.com/120437
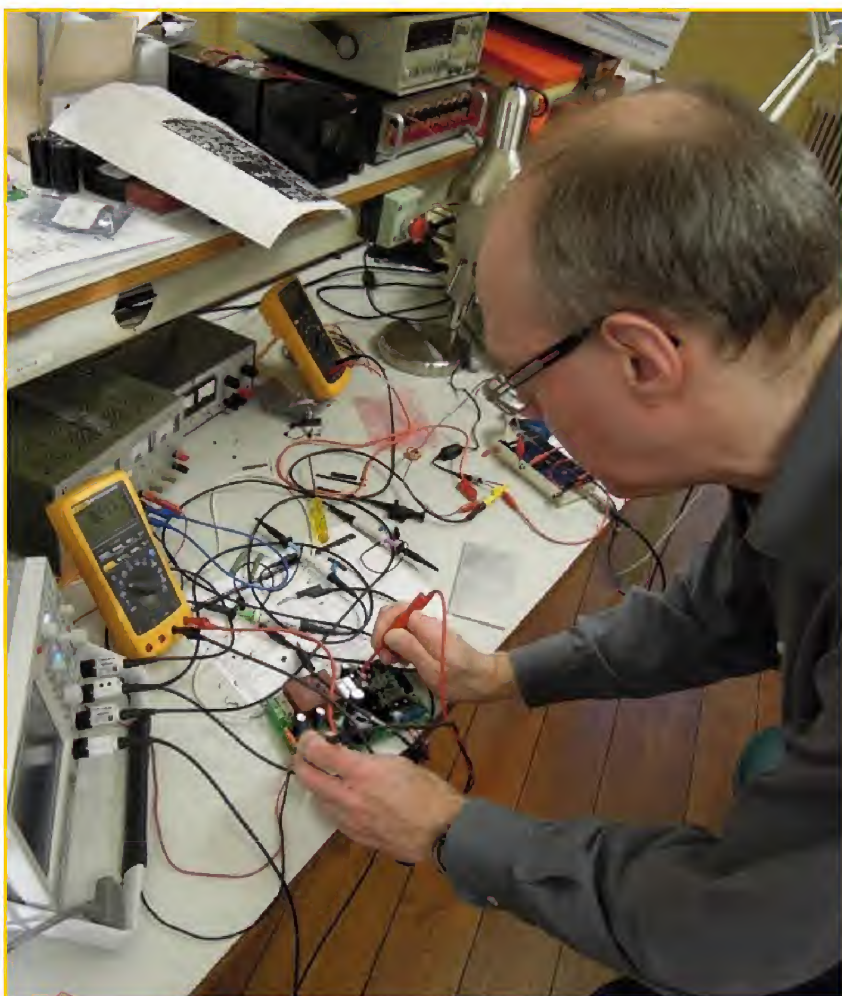
Up, Up and Away in my Beautiful Balloon

We all know that our world has three dimensions. If you add time then you get to four dimensions. But what would the fifth dimension look like? Mathematicians and physicists have been working on this for years, they even went up to 26 dimensions if I am not mistaken, but for us mortals this is a little too abstract. And yet, ever since 1967 the fifth dimension has been right in front of our eyes! Or should I say ears? It was 1967 that an American band called The 5th Dimension launched the song *Up, Up and Away* that won five Grammy Awards. And who wrote that song? Indeed, Jimmy Webb. Need I say more? The fifth dimension is simply the *world-wide jimmy web*. Not convinced? OK, go to the Elektor dot LABS website and what do you see? A proposal posted by *ale* for a weather balloon controller project. Balloon, web, 5th dimension, it all falls into place, doesn't it?

Have a look at ale's weather or, as he called it, Sounding Balloon Main Unit Controller, it is an interesting project. He proposes a system capable of transmitting six analog signals plus position data over an FM radio link. These channels can be used for sending back to earth parameters like temperature, infrared and UV light data, battery level (although I wouldn't really care about that) or atmospheric pressure. Since it has to go up and away in a beautiful balloon, the system has to be cheap and lightweight.

And now that we're on the subject of ballooning, did you know that Elektor published the world's first Near-Field Communication (NFC) enabled book? This beautiful book is about ballooning and it contains links up and away to the web. It is the world's first book with five dimensions.
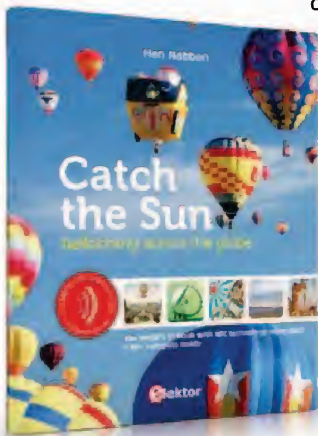
Enter the 5th dimension:

www.elektor-projects.com/9121102594

Editor's Choice

A number of .LABS projects have been selected by our editors and should be published in the near future. For some of these projects, sadly we found that the **original poster** (**OP**) does not reply to our messages. Therefore, if you posted a project, please check on a regular basis the email account you used to access .LABS. We will not get you in publication if we cannot get in contact with you.

Here is a selection of projects we think is interesting and which we would like to publish in the printed magazine.

Laser TV

The laser TV project by OP hpt is an attempt to project a video image by means of 30 rotating mirrors mounted on a VHS head motor. Initial trials have been done. The main challenge is the phase synchronization of the top plate, and the OP is looking for interested BASCOM programmers to develop the motor PLL (or a similar software solution).

www.elektor-projects.com/9120502218

Analog Theremin

The Theremin was one of the first electronic musical instruments. It is played by moving your hands close to two whip antennas. Simple digital square wave based designs are all over the Internet, but OP *D.Philips* would like to build an analog Theremin with real sine waves and high-quality controls. Can you help him?

www.elektor-projects.com/9120502157l

Waterbed energy saver

Do you have a waterbed? OP Schwabix does, but he feels that it consumes too much energy. His idea is to turn off the heating of the waterbed during daytime, when the bed is not used. Of course the heating should be switched back on early enough to make sure that the bed will have a convenient temperature when Schwabix dives into it. The OP would like of course assistance and fruitful discussions.

www.elektor-projects.com/9121102686

Portable Radio/Media Player

OP *rsavas* has designed a product he calls the Portable Radio/Media Player. It is a radio, an MP3/AAC/WMA player (USB memory stick or SD memory card) and also connects to a PC using the USB codec (PC Stereo/sound card). It has a source selector, volume control, headphone amp, and a 20-watts Class-D amp. Since its features are only limited by software the OP is looking for programmers to help him finish this project.

www.elektor-projects.com/9120502222

www.elektor-labs.com

Component Tips

By Raymond Vermeulen (Elektor Labs)

MIC502 and MAX6643 Fan-Speed Controllers

For this month's installments I researched two ICs that provide stand-alone functionality as fan-speed controllers. This can be very handy when you have a system which is dissipating a lot of power, but is also built into some kind of enclosure. This makes forced-cooling with a fan a necessity. If you would like to make the control of this fan dependent on the temperature, but don't want to delegate this task to the existing microcontroller, then a separate control IC will be very useful. You only need to connect a sensor and a fan. One striking detail is that the ICs presented here do not follow the standard 4-pin, 25 kHz PWM fan speed control, but switch the power supply voltage directly at much lower frequencies.

(120569)

MIC502

The MIC502 made by Micrel uses an NTC or PTC as the temperature sensor, with the option of having a second sensor. The sensors are connected to inputs VT1 and VT2. A voltage from about 30% to 70% of V_{dd} produces a duty cycle of 0% to 100%. The highest of the two inputs takes precedence, which is a very nice feature if you use one sensor to measure the ambient temperature and use another to monitor the temperature of a component that is likely to get hot. A voltage can be applied to the V_{slp} pin, which causes the chip to go into a sleep state when both inputs VT1 and VT2 drop below this value. When either VT1 or VT2 go above this value then the IC will be reactivated. This is mainly to avoid the fan stalling at a duty cycle that is too low for it to operate properly. A timing capacitor is connected to pin CF, a value of 100 nF is recommended for a frequency of 30 Hz. You can, however, also set a higher frequency. According to the data sheet the range is from 15 to 90 Hz.
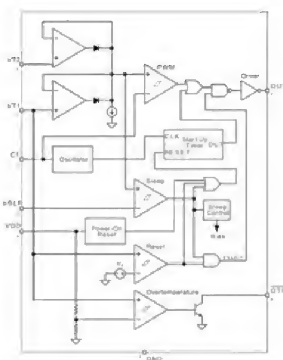
Figure 1. Block diagram of the MIC502.
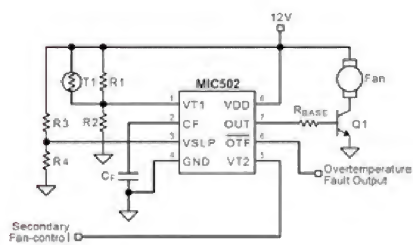
Figure 2. Application example using the MIC502.

MIC502 datasheet: www.micrel.com/_PDF/mic502.pdf

MAX6643

The MAX6643 uses a diode-connected transistor to do the temperature measurement. This IC has three different temperature control settings. The temperature is set (between 60 and 100 °C) with the 'Overtemperature Threshold' inputs (pins OT1 and OT2) and the OT-output will be activated when this temperature is exceeded. The 'High temperature Threshold' inputs (pins TH1 and TH2) determine at which temperature the duty-cycle of the PWM-signal increments by one step. The 'Low temperature Threshold' inputs (pins TL1 and TL2) determine the temperature at which the duty cycle of the PWM signal goes down by one step. For each of these temperature-thresholds a number of different values can be set by connecting the corresponding pins to V_{dd} or GND or leave open (refer datasheet).

There is also a 'FULLSPD' pin, to force a duty cycle of 100%. This can be useful when an excessive temperature has been detected. In addition there is an input for the tacho signal from the fan, which allows the 'FANFAIL' pin to indicate when there is a problem with the fan.
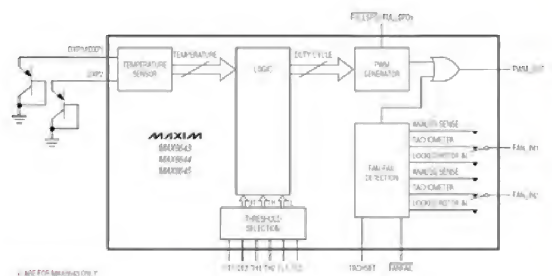
Figure 3. Block diagram of the MAX6643.
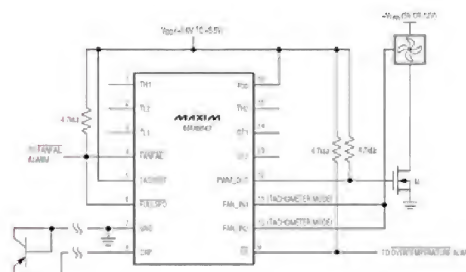
Figure 4. Application example using the MAX6643.

MAX6643 datasheet: http://datasheets.maximintegrated.com/en/ds/MAX6643-MAX6645.pdf

Audio Streamer,
testing one two

By Thijs Beckers
(Elektor UK Editorial)
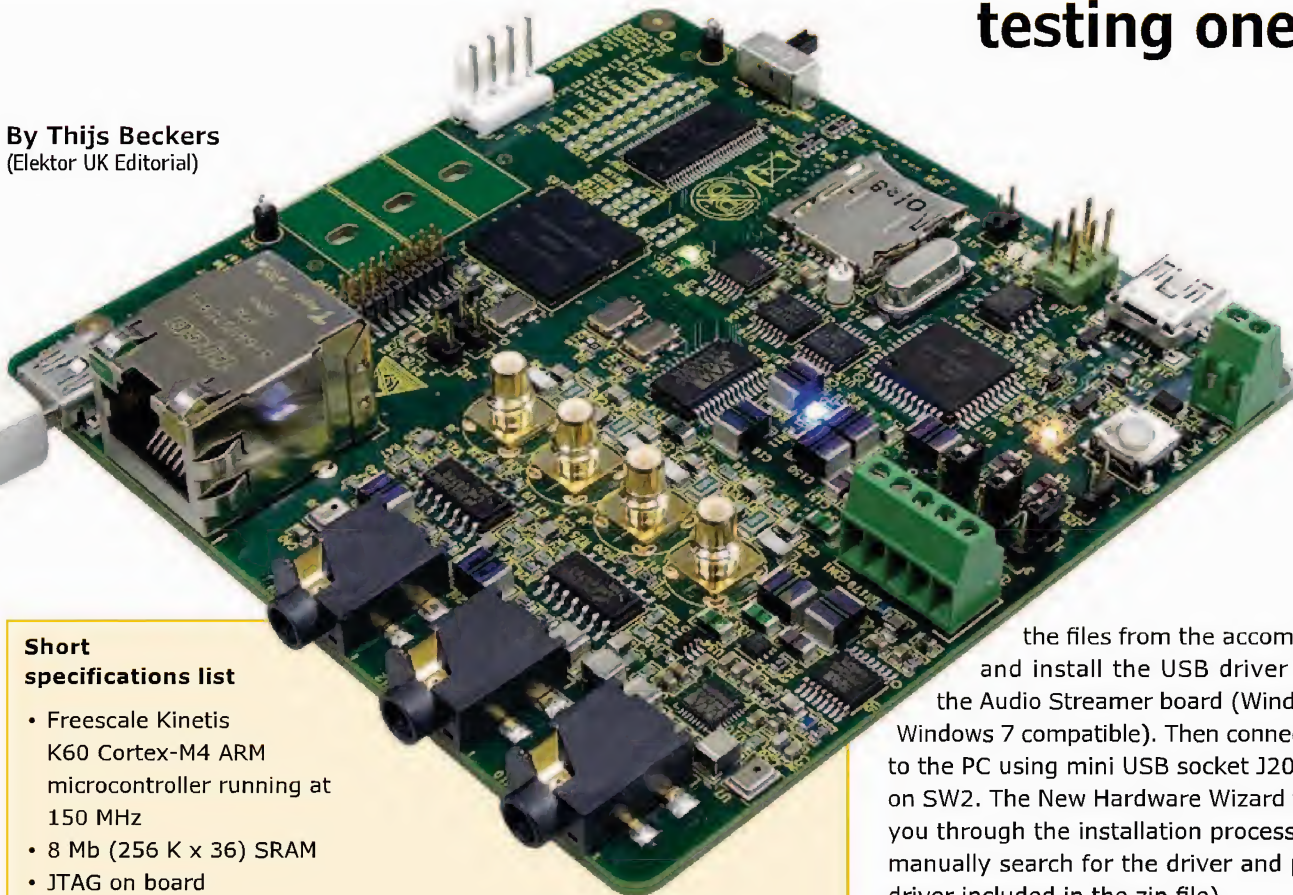
Short specifications list

- Freescale Kinetis K60 Cortex-M4 ARM microcontroller running at 150 MHz
- 8 Mb (256 K x 36) SRAM
- JTAG on board
- Fits on Future Board
- SD Card slot
- 3 Touch Sense buttons
- DACs: WM8524 (low cost) and WM8741 (high-end)
- High-end Nichicon multilayer polymer film capacitors
- 160-way Micro-Blox interface
- Supported formats: WAV, MP3, FLAC [3] and Ogg Vorbis [4] (licensing issues prohibited the support of Apple formats).
- Ultra Low Power CODEC: WM8904 [2]

Recently we received one of Future Electronics' latest development kits, the Audio Streamer [1] to play around with. This Micro-Blox family member is a 'rapid proof-of-concept development system for OEMs', claimed to provide designers with a fast and comfortable way of implementing new applications with the latest generation of digital audio DACs and CODECs from Wolfson Microelectronics [2]. A short overview of the specs is provided in the text inset.

The board set up is quick and easy. First, unzip the files from the accompanying CD and install the USB driver supporting the Audio Streamer board (Windows XP and Windows 7 compatible). Then connect the board to the PC using mini USB socket J20 and switch on SW2. The New Hardware Wizard then guides you through the installation process (choose to manually search for the driver and point to the driver included in the zip file).

Once installed, audio can be streamed via the USB connection or via a network connection. A comprehensive description for setting up the network connection is included in the Quick start guide. A basic GUI, called AudioDemo, is provided for configuring and playing around with some of the options of the board.

After choosing your connection method, a drop down menu, accessible under the *Configure* tab, enables you choose the DAC or CODEC to be used for playing back the audio stream. Several sub-settings are available. Depending on the selected DAC or CODEC, these comprise the output selection (Line Out or Headphones), a five band parametric equalizer, several filter response curves, anti-clipping mode, soft mute and the source of the Master clock: the Kinetis K60 or an external oscillator.

Under the *Play* tab a library of preloaded files is shown. Here you can add your own test files (on your PC or on the SD Card in the slot). Two sliders provide control over the volume and balance of

the output signal.

The board is also able to record in WAV format (saved to the connected PC). For this, two sound sources are implemented on the board: a stereo digital silicon microphone (two WM7210s set up in stereo) and an analog silicon microphone (WM7110). Both sound sources are fed into the WM8904 low cost CODEC. This provides a basic test platform for all kinds of applications, such as security and intercom communications.

Aimed at HiFi applications, the High-end WM8741 DAC from Wolfson Microelectronics provides an excellent way to develop SD, USB and/or Ethernet streaming applications. SMC connectors J5-8, provide balanced line out signals directly from the DAC, allowing for true measurement of the S/NR between the DAC and the op amps (the S/NR could be enhanced with the use of filtering around the op amps). Special high grade multilayer polymer film capacitors are used in the audio circuits. For applications where HiFi quality is not required the WM8524 DAC is offered as an alternative solution.

The complex decoding of FLAC and Ogg Vorbis streams only needs about 30 to 40% computing capacity of the Kinetis Chip, so there's lots of headroom to implement parallel running applications. In case you only need MP3 decoding, which is far less demanding in terms of computing power, a smaller Kinetis microcontroller could be chosen and the SRAM could be omitted in the end application.

To start developing with the Audio Streamer, IAR Embedded Workbench for ARM is the suggested environment (version 6.3 or higher). The source code of the AudioDemo application, written in C#, is freely available from Future Electronics. The board uses the MQX real time operating system, for which Freescale's MQX Software Solutions offers the developer everything he or she needs. There are lots of other features not implemented in the AudioDemo application, so there's still a lot to be explored. In conclusion we can confirm that the Audio Streamer is an uncomplicated way of getting a streamer into the hands of a developer. Using the supplied documentation and (freely) provided software, a complete streaming application is set up very quickly and, eliminating the need for developing a hardware prototype for the (software) developer to get started, the process of developing a properly functioning end application should be speeded up significantly.

(120699)
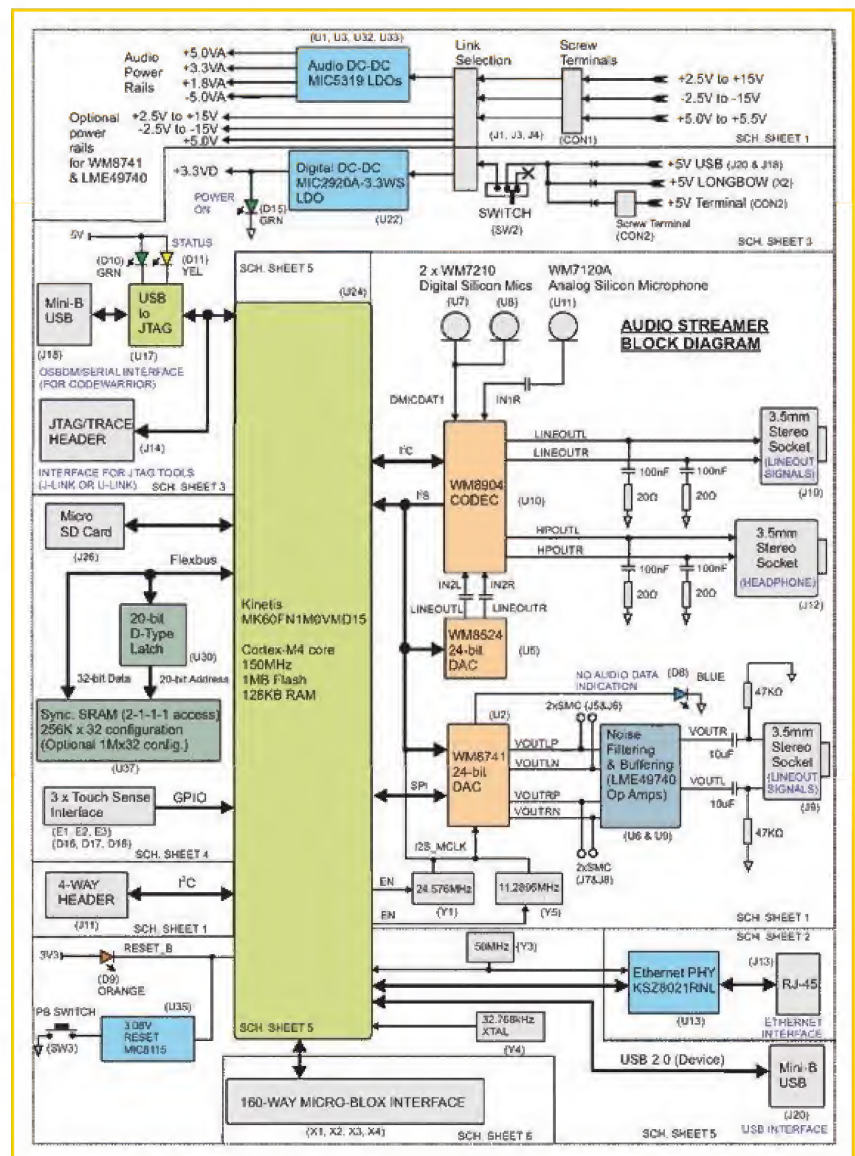
Internet Links

[1] www.my-boardclub.com/future_blox/
miniblox.php?blox=32

[2] www.wolfsonmicro.com

[3] http://en.wikipedia.org/wiki/FLAC

[4] http://en.wikipedia.org/wiki/Vorbis
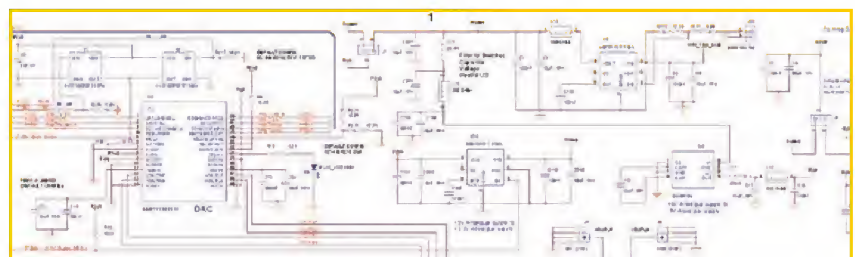
Toroid cutting

By **Thijs Beckers**
(Elektor Editorial & Labs)

Elsewhere in this edition we described an inrush current limiter which uses a specially adapted ferrite toroid. Here's how we prepared the toroid for deployment in the circuit.

(120717)

First pick the correct ferrite toroid for your application, put the windings on and secure it in a vice (do not over tighten). We used small sheets of aluminum — a soft metal, to prevent damaging the toroid.

Using a Dremel or similar tool equipped with a cut-off disc suitable for cutting metal, start cutting at a right angle w.r.t. the toroid. Safety glasses are required, small pieces can chip from the ferrite as well as from the disc.

Go slow! It took us about 3 to 4 minutes to get this far. The rpm should be quite high, but not exceeding the disc's maximum rated rpm.

Adjust the toroid so that your second perpendicular cut can be made easily. Be careful: the toroid loses its rigidity when cut, and the material is quite brittle.

Take your time and be sure not to force your way through, as the cutting disc is not suitable for grinding and prone to breaking when pressure is applied perpendicular to the disc radius.

Done! Be careful, the toroid may be hot from the cutting. Again, be gentle while handling the cut toroid, as it's quite brittle and breaks a lot easier than when in its closed toroid form.

USB Current Unlimited... the sequel

By
Raymond Vermeulen
(Elektor Labs)

A quick recap: a few months ago in our September 2012 edition I invited you to share your knowledge on how and if USB current is limited. In the following months I received plenty of replies from you and a perception started to form. It seems that reality has more pitfalls than I assumed.

At the host side, central current sourcing circuits are often limited at well over 500 mA. For example: on a PC mainboard with 10 USB ports it is not uncommon that a single 5A current limiter is 'protecting' all ports.
Several respondents have tested a variety of devices as well, including loads consuming 700 mA when idle and over 1 A when active. There were some arguments that the limit was intended for unpowered HUBs.
Some interesting contributions:

"The disparity between the 500 mA allowable current (chosen because it allows classification as toys) and the 100 mA initial current budget in USB is for unpowered hubs. Since the hub is allotted 100 mA, 400 mA is left to share on its downstream ports; which explains why they have 4 ports. A high power device will be denied when placed behind an unpowered hub. Sadly, manufacturers consistently skimp on parts they believe non-essential, so you may often find devices with no current limit on both sides and hubs that cannot detect if they are powered or not." — Yann Vernier.

"USB is not designed to limit current based upon device request. What actually happens is that a USB host is supposed to have a driver that is aware of the total current capabilities of each port. ... A very important consideration here is that nothing physically enforces this 100 mA limit — certainly not the USB host hardware. ... Have a look at http://goo.gl/9vjD7" — Ian G3ZHX

"Like you, we built a current-shunt adjustable from 10 mA to 2 A that can be paralleled to a low-power USB device like a mouse. Our measurements concluded that 500 mA is supplied without any negotiations between the device and the host. We could up the current to about 1.4 A before a limiter activated and the USB hardware shut down. USB devices should therefore monitor their own current consumption and make sure it doesn't exceed 500 mA. A short-circuit is also recognized well after 1.4 A is exceeded.

Furthermore we have tested several USB Devices. 'Lide'-series USB scanners from Canon for example use about 700 mA when idle and raise this number to a whopping 950 mA when scanning.

Not very USB regulation compliant.
Inserting a Hub (of course self powered, using its own wall wart) provides no solution either. With one of the tests we plugged the mouse-shunt circuit in a similar hub and no limiting at 500 mA occurred. Even worse: when we unplugged the hub's power source, the hub drew its power directly from the host (the PC or laptop). Setting up some sort of buffering using a hub therefore proved impracticable." — Gerald E. Riemer (Dipl. Phys. Ing.)

The People have Spoken.
Thank You and Good Night!

(120575)

Aviation Scanner

For aircraft enthusiasts and in particular plane spotters, it's always interesting to listen to communications between aircraft and control towers. This receiver makes it easy to listen to all traffic in the civil aviation band. It is easy to build, has only one hardware adjustment, and can even be controlled from a PC with a USB port.

By **Gert Baars**
(Netherlands)

"Boston Approach, Delta603 with information delta, descending to flight level 70 just passed LWMr inbound, speed 250 knots..."

It's always interesting to listen to communications between aircraft and ground control. Even though this usually consists of standard information about position or instructions to pilots for take-off or landing, it gives you a bit of a thrill – it's almost like sitting in the cockpit next to the pilot. Along with a lot of hobbyists who listen to aviation communications because they find this fascinating, plane spotters like to listen so they can find out which aircraft are arriving or getting ready for departure.

Of course, ready-made aviation scanners are available at fairly reasonable prices, but it is still interesting to make your own and play around with it. The scanner described here has a very simple design, but it offers a large number of features thanks to the integrated microcontroller, which also allows you to control the scanner and program the frequencies from a computer.

Considerations

To keep the circuit reasonably simple, the scanner is limited to the civil aviation band (108–137 MHz). The civil aviation band was originally divided into channels spaced at 25 kHz, which was later updated to 8.33 kHz (25/3). In practice, this scheme is rarely used. The scanner described here can be tuned with a resolution of 1 kHz, so it can never differ from the actual frequency by more than 330 Hz. The desired bandwidth for AM signals is approximately 6 kHz. All in all, this determines the minimum requirements for the scanner.

To avoid the need for implementing a keypad and a display, we opted to replace these components by a terminal. The scanner communicates over a serial link (USB) to a PC, which displays a control interface using HyperTerminal. You can use the PC to program the frequencies and set various parameters.

For the receiver portion we chose a relatively simple approach with a single IC — a single-conversion superheterodyne FM receiver that is employed as an AM receiver by using the signal strength detector as an AM detector. In a single-conversion superheterodyne receiver there are

Portable scanner with USB interface

image frequencies that must be suppressed. This is easier if the distance between the receive frequency and the image frequency is relatively large, which means using a high intermediate frequency (IF). The highest possible IF with the selected receiver IC is approximately 25 MHz. Although 27 MHz is within the usable range, filters for this frequency are usually expensive and hard to obtain. Here we opted for two simple ladder filters with 27 MHz crystals — in total four crystals costing about one $1.00 each. The advantage of a 27 MHz IF with the LO frequency below the receive frequency is that the image frequencies lie in the range of 54 to 83 MHz, just below the VHF FM broadcast band.

Digital tuning in receivers is often implemented using a PLL or DDS, but to keep things as simple as possible we looked for a simpler solution for this project. One option is a frequency locked loop (FLL). With this approach the microcontroller measures the frequency of the VFO, compares it to the desired frequency in the software, and adjusts it using a DAC. From a few tests we concluded that this works fairly well if the DAC has sufficient resolution to
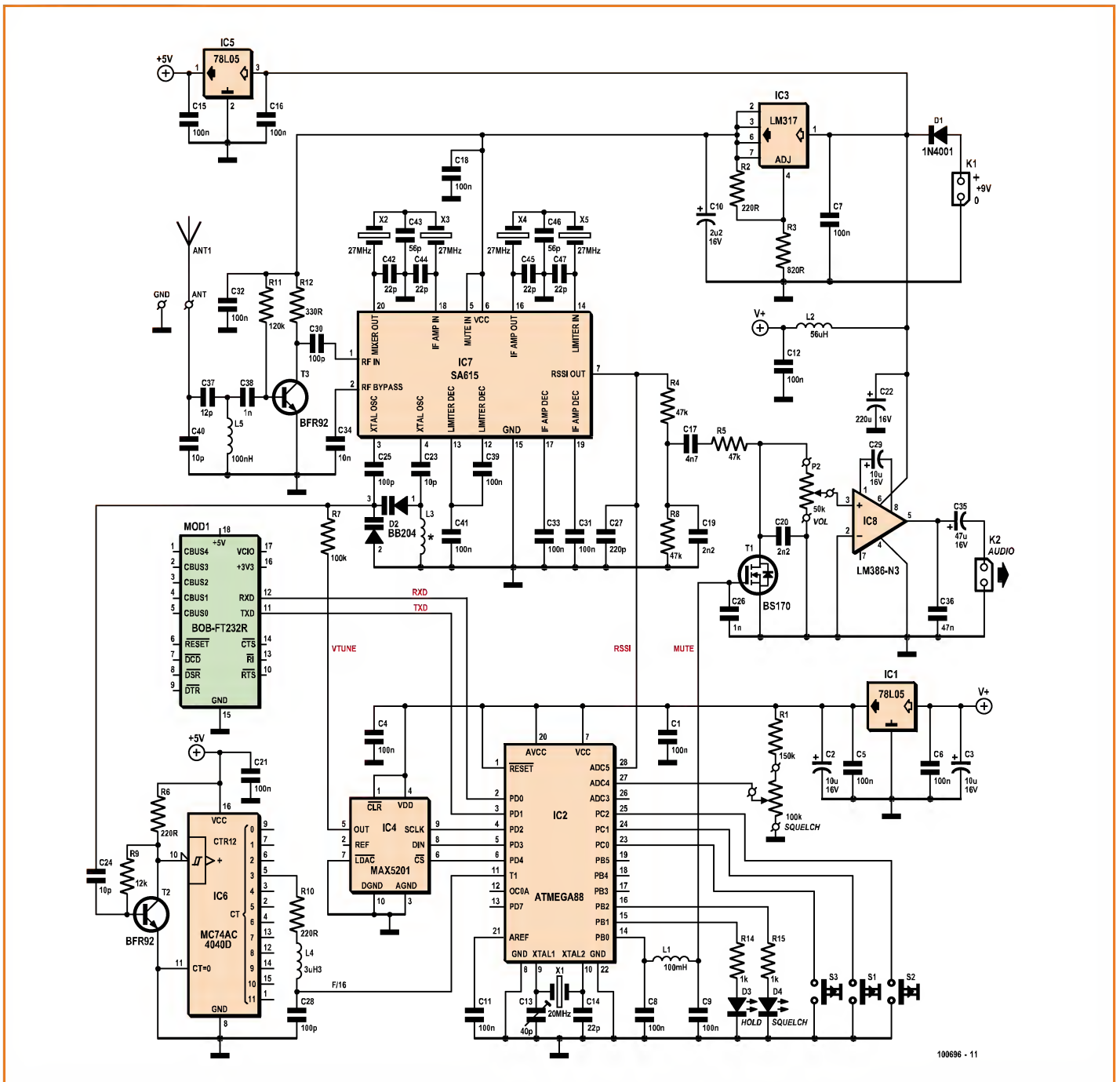
Figure 1.
Schematic diagram of the receiver, with the analog portion at the top and the digital portion at the bottom. The three voltage regulator ICs provide stable supply voltages for the various parts of the circuit.

enable small frequency adjustments. The counter input of the microcontroller cannot handle very high frequencies, so we inserted a frequency divider between the oscillator and the microcontroller. This yields a circuit with relatively few (and simple) components. The scanner has just three buttons, a volume control and a squelch control. All you need to make the scanner portable is to program the frequencies and settings with a PC.

As an extra feature, the scanner has an operating mode in which it automatically searches for frequencies and programs them in memory. This involves setting the squelch level, starting the scanner in search mode, and waiting until the search is completed. In this mode the scanner automatically scans the aviation band, listening on each frequency in turn and storing in its memory each frequency where it finds a signal.

Receiver portion

The receiver (see **Figure 1**) is built around an NE615 (SA615) IC. This IC is an improved version of a combination of the NE602 (mixer and VFO) and the NE604 (IF amplifier, limiter and FM detector). FM detection is not relevant here, so some of the pins of the IC are not used. However, the IC has a receive signal strength indicator (RSSI) output, which provides a voltage proportional to the signal strength. This signal is split into two: one for the audio output (with narrow filtering for noise suppression), and another for signal detection by the microcontroller. This allows the microcontroller to stop scanning if it detects an active signal.

To improve reception, the signal from the antenna is filtered and amplified before being fed to the receiver IC. There it is mixed with the signal from the VFO, which is tuned using a dual varicap. This results in an IF signal at 27 MHz, which is filtered by a pair of two-element crystal ladder filters (X2–X5) with a bandwidth of approximately 6 kHz. The IF is not exactly 27 MHz, but instead 26.998 MHz. This can be changed in the setup parameters, but the default IF setting is 26.998 MHz. The crystals must be identical fundamental-frequency types, but a deviation of a few kHz is tolerable and can be handled by adjusting the setup parameters if necessary.

Do not use third-overtone (9-MHz fundamental) crystals.

The VFO signal is fed via a gain stage (T2) to a 74AC4040 (IC6) wired as a divide-by-16 stage. The VFO frequency range is 81 to 110 MHz. The upper frequency limit of the microcontroller's counter input is 8 MHz, so a divisor value of at least 14 is required.

The audio output signal from IC7 (pin 7) is fed to the volume control P2 via several resistors and the mute FET T1. The volume control passes the signal to a small audio amplifier IC (IC8, an LM386), which can directly drive a small 4-ohm or 8-ohm loudspeaker.

The combination of analog (RF) and digital signals makes extensive supply voltage regulation necessary. A total of three voltage regulators are used for this purpose. IC3 provides a clean 6 V supply voltage for the entire RF portion. The digital portion has its own 5 V regulator in the form of IC1, and IC6 has a separate supply voltage provided by IC5 (a 78L05).

Microcontroller portion

The ATmega microcontroller (IC2) has to look after several tasks. It measures the frequency of the VCO divided by 16 by means of a counter with a gate time of 16 ms, yielding measured values in kilohertz. The VCO is driven by a 16-bit DAC, type MAX5201 (IC4). This IC was chosen due to the required number of bits. With a frequency range of 29,000 kHz and 2^{16} steps, the resolution is approximately 440 Hz. This is acceptable for adequately precise tuning with an IF bandwidth of 6 kHz. Two other significant reasons for choosing this device are its rail-to-rail output amplifier and its serial control interface.

The microcontroller has enough EEPROM memory to store more than 100 frequencies in addition to various settings. We opted for 100 frequencies, which is more than necessary, but you can also select frequency bins, such as 5 out of 20 frequencies.

Two ADC inputs of the microcontroller are used to measure the signal strength and the setting of the squelch potentiometer (pins 28 and 27). These values are compared to internal reference values, and based on the results scanning is either stopped or continued. Two LEDs (D2 and D4) provide information on aspects such as the squelch threshold and the wait time during scanning. There are three pushbuttons (S1–S3), which can be used for functions such as blocking a channel during scanning.

The USB-FT232R breakout board [1], which has been used in several previous Elektor projects, is connected to pins 2 and 3 of the microcontroller. Space for this module is reserved on the PCB. This converter allows the scanner to be connected to a PC via a USB cable.
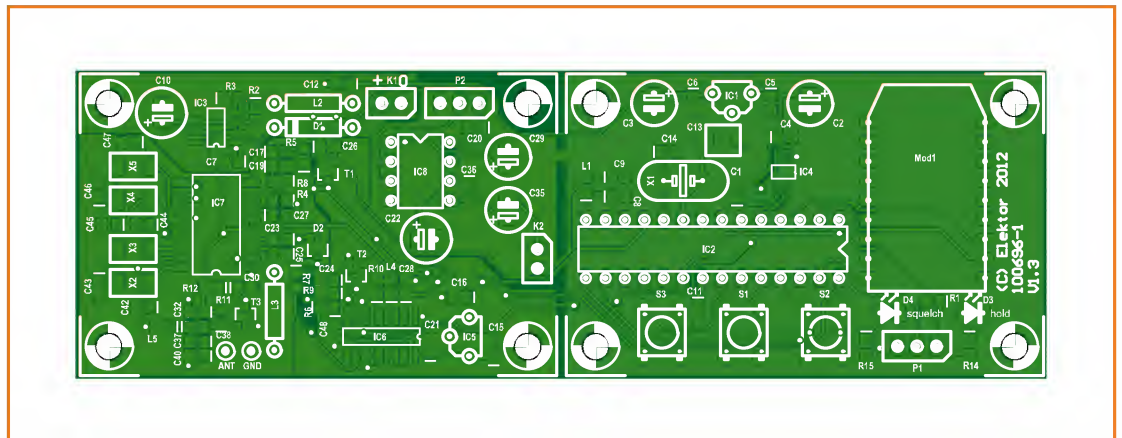
Figure 2.
The PCB consists of two sections that can be separated if desired, depending on the chosen enclosure.

Software

The software enables the hardware of the scanner to do its job properly. The main loop reads frequencies sequentially from the EEPROM. For each frequency, the IF must be subtracted to determine the VFO frequency. Setting the frequency directly is not possible here, due to the FLL operating principle. For this reason, the frequency is set using the successive approximation method. With a 16-bit DAC, this requires 16 measurements with a duration of 16 ms each. Although this makes scanning slower, it allows the scanner design to be simpler than with a PLL or a DDS IC.

Once the frequency has been set, the software checks the RSSI output to see whether a signal is preset. If it is, the mute output is set low and the audio signal is sent to the loudspeaker. If the signal goes away, the frequency is monitored for a configurable time interval, and after this time scanning is resumed by reading a new frequency from the EEPROM and setting the VFO frequency. The frequency is held constant during the waiting interval and while listening on a particular frequency. The serial buffer and the states of the pushbuttons are also read in during scanning. This allows the input from the terminal to be read, and if button S1 is pressed the channel is skipped and the frequency is marked as 'B' in the EEPROM. This blocks the channel, which may be desirable if the scanner keeps getting stuck on this channel, perhaps due to the presence of a carrier wave. This blocking can be cancelled from the terminal.

The scanner has a number of default settings (such as the IF value), which are read from program memory in EEPROM when the scanner starts up. This allows these parameters to be configured, either directly in the scanner or by using the terminal.
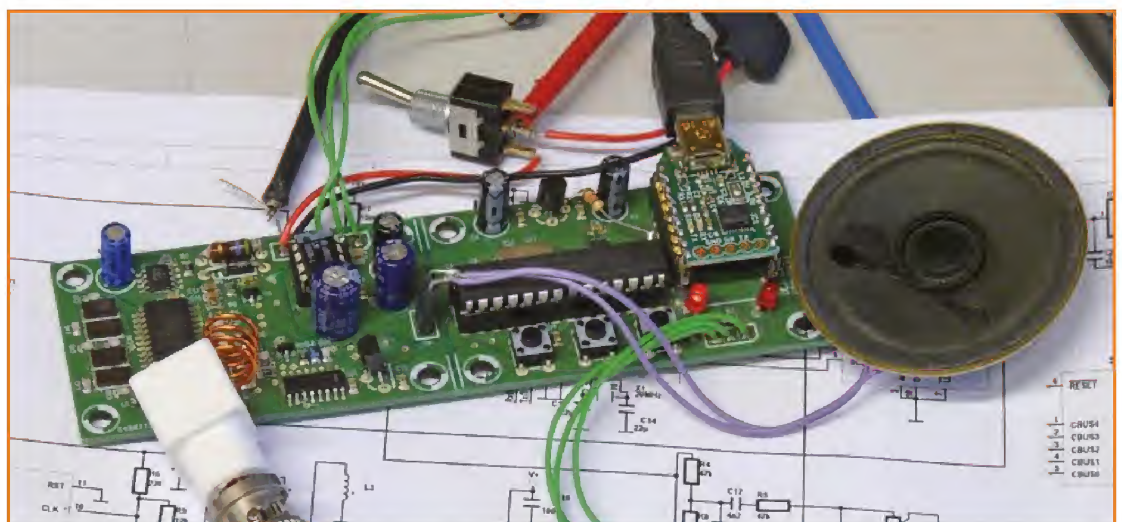
Figure 3.
The prototype is a bit on the scruffy side, but it works just fine.

Construction

Figure 2 shows the PCB layout designed for the aviation scanner. It is divided into an analog section and a digital section, which can be separated if necessary, depending on how you fit them in an enclosure. If you separate the two sections, you will have to join them together with a few wires (five wire links plus a ground line between the two sections).

It's best to start by fitting the SMDs, since this is the most difficult task. Once that's done, you can fit and solder the IC sockets and the ordinary components. The two potentiometers are mounted on the enclosure and connected to the PCB by a few short pieces of insulated wire. Coil L5 is the only coil that you have to wind yourself. To do this, wind four turns of #18 AWG (diam. approx. 1 mm) silver-plated copper wire on the smooth shank of a 7-mm drill bit (old US letter code: J). Then stretch this coil to an overall length of approximately 7 mm and solder it to the PCB. Cut a bit of styrofoam to size and insert it into the coil to prevent vibration. Finally, fit the pre-programmed PIC, the LM386 and the USB-FT232R breakout board on the PCB.

When selecting an enclosure, bear in mind that in addition to space for the PCB you need space for the potentiometers, a small loudspeaker, and a 9-V battery or battery pack. The buttons must be externally accessible (one option is to fit them with small tubular extensions), and the LEDs must be visible (mount them raised above the PCB if necessary). You also need to mount a BNC connector for the antenna.

For outdoor use, you can connect a whip antenna with a length of about 60 cm (approx. 2 feet). For indoor use you can connect a more effective antenna, such as the one described elsewhere in this edition.

Adjustment

There is a 40 pF trimmer capacitor next to the clock crystal for the microcontroller, which allows the clock frequency to be set to exactly 20.0000 MHz to provide accurate tuning. Set the trimmer to the midrange position to start with. It can be adjusted by measuring the frequency at the output of the 74AC4040 and using the 'T' command to set the scanner to a fixed frequency. This frequency, less the IF and divided by 16, should appear at the divider output. After this frequency has been set exactly,

COMPONENT LIST

Resistors
(shape: SMD 0805)
R1 = 150kΩ
R2,R6,R10 = 220Ω
R3 = 820Ω
R4,R5,R8 = 47kΩ
R7 = 100kΩ
R9 = 12kΩ
R11 = 120kΩ
R12 = 330Ω
R14,R15 = 1kΩ
P1 = 100kΩ potentiometer, linear
P2 = 50kΩ potentiometer, logarithmic

Capacitors
(default shape: SMD 0805)
C1,C4–C9,C11,C12,C15,C16,C18,C21,C31,C32,C33,C39,C41,C48 = 100nF
C2,C3,C29 = 10µF 35V radial
C10 = 2.2µF 63V radial
C14,C42,C44,C45,C47 = 22pF
C17 = 4.7nF
C19,C20 = 2.2nF
C22 = 220µF 16V radial
C23,C24,C40 = 10pF
C25,C28,C30 = 100pF
C26,C38 = 1nF
C27 = 220pF
C34 = 10nF
C35 = 47µF 16V radial
C36 = 47nF
C37 = 12pF
C43,C46 = 56pF
C13 = 8.5pF – 40pF trimmer (Murata TZB4P400AB10R00)

Inductors
L1,L5 = 10nH (SMD 0805)
L2 = 56µH
L3 = 4 turns 18 AWG silver plated, length = 8mm, inner diam. 7mm
L4 = 3.3µH (SMD 0805)

Semiconductors
D1 = 1N4001
D2 = BB207 (TO236)
D3,D4 = LED, low-current, 3mm
T1 = BS170F (SOT23)
T2,T3 = BFR92A (SOT23)
IC1,IC5 = 78L05
IC2 = ATMEGA88-20PU, programmed, Elektor # 100969-41
IC3 = LM317 (SOIC-8)
IC4 = MAX5201AEUB+
IC6 = 74AC4040 (SOIC-16)
IC7 = SA615D (SOIC-20)
IC8 = LM386 (DIL-8)

Miscellaneous
X1 = 20MHz quartz crystal
X2–X5 = 27MHz quartz crystal, fundamental resonance, (e.g. Citizen CS1027.000MABJ-UT)
USB-FT232R "BOB" breakout board, Elektor # 110553-91 [1]
S1,S2,S3 = 6mm pushbutton (Multicomp MC32830)
2 pcs 2-pin pinheader
28-pin DIL socket for IC2
8-pin DIL socket for IC8
2 pcs 10-pin SIL socket, 0.1'' pitch (for USB-FT232R "BOB" breakout board)
Miniature loudspeaker, 4 or 8 Ω
PCB # 100696-1

Operation

The following screen is sent to the terminal when the scanner is switched on:

```
****************** Airband scanner startup ******************

Command key functions:

H          -help (this page)
S          -show stored frequencies
B <fn>     -block on/off, fn is freq. number
T F(6)     -tune to frequency(kHz)
R          -scanner run/stop
F fn F(6)  -store frequency(kHz) fn = freq. no.
I Fif(5)   -IF frequency (kHz)
N <nf>     -#frequencies to scan (1-100)
W <Tw>     -scanning wait time (x 0.5 s)
M <fn>     -memory start from freq. no. fn
P <nr>     -#passes (2-15) for autom. search

current settings:

Fif       = 26998
Twait     = 2
#Freqs.   = 20
Mem.start = 1
#Srch.pas = 5

Fn 1
```

At this point the scanner is in scan mode and the frequency number is shown at the bottom of the screen. In order to enter commands, first type 'R <Enter>'. This stops the scanner and allows you to enter single-letter commands. Most of them are self-explanatory after you have tried them once. The command 'S <Enter>' causes the frequencies to be shown in a table. The first and last frequencies in the table can be set with the 'M' and 'N' commands, respectively. These settings also determine the start and end frequencies used for scanning. You can additionally define frequency bins with these parameters. In this case a frequency bin configured by M and N is scanned (for example, frequencies 1 to 20). The bin range can be changed using the terminal, for example frequencies 40 to 59. This allows the 100 programmable frequencies to be divided into bins.

The scanner can also acquire frequencies itself. For example, if you have entered a number of frequencies (such as 1 to 40) use the 'T' command to set a receive frequency and then check whether the IF value in the setup parameters is correct. This can be adjusted with the 'I' command if necessary.

The tuning range can be checked by using the 'T' command to set the frequency to 108.000 and 137.000. If the frequency shown on the monitor is too high, the VFO coil (L2) must be shortened a bit by pinching it slightly. If the frequency is too low, the coil must be lengthened a bit by stretching it.

The frequencies shown on the terminal should be exactly the same as the frequencies set with the 'T' command.

The source code and hex code files for the microcontroller can be downloaded free of charge from the project page for this design [2]. A pre-

that are not allowed to be overwritten, you can set M to 40 and N to 40. The squelch potentiometer must be set properly for normal use.

If you hold button S2 pressed while switching on the scanner, the Hold LED (D3) will light up. The scanner will then scan 1160 channels (corresponding to (137,000 - 108,000)/25) several times. The number of times can be set with the 'P' command. At the end of the search, the frequencies that were found are programmed as frequency numbers 40 to 80. After this LED D3 goes dark. The result can be viewed with the 'S' command after restarting the scanner. In a test at approximately 30 miles from an airport with P set to 6 and 20 frequencies, virtually all frequencies (including tower, approach, departure, radar and other commonly encountered frequencies) were found automatically in half an hour. The frequencies are automatically sorted by the number of times they are found, with the one found the most in the first memory location, the next one in the second location 2, and so on.

The scanner can also be used from the terminal as a normal receiver with the 'T' command. For example, entering 'T 122400' tunes the receiver to 122.400 MHz, and this frequency is shown on the terminal along with the signal strength (1–9).

Modes

Scanner

The scanner begins scanning from memory after it is started up. The Squelch LED (D4) lights up when the signal exceeds the squelch threshold, and LED D3 light up when the scanner stops to listen to a signal. If the signal goes away, LED D4 goes out, but you can continue to listen during the wait time. LED D3 goes out only after the wait time has expired and scanning starts again. The frequency can be 'frozen' by turning the squelch threshold all the way down.

The frequency number in memory is constantly sent to the terminal during scanning. If a continuous signal is found during scanning and scanning stops, the frequency can be blocked by pressing button S3. This marks the frequency in the memory with a 'B', which can only be deleted from the terminal.

Receiver

If button S3 pressed and held for at least 1 second after the scanner has been started, the scanner enters Receiver mode. The receiver can be tuned with buttons S1 and S2 (down and up, respectively), and the squelch control works in the normal manner. LED D3 indicates whether the minimum or maximum frequency has been reached. The receive frequency is initially set to the minimum frequency when the scanner is started in Receiver mode. Button S3 acts as 'next' button in this mode, causing the receiver to search with increasing frequency until it finds a signal that exceeds the squelch threshold. When it does, it stops searching and LED D3 lights up. When S1 is pressed again, LED D3 goes out and the receiver searches for the next signal. When it reaches the maximum frequency, it continues searching from the minimum frequency.

Search mode

If button S2 pressed and held for at least 1 second after the scanner has been started, LED D3 lights up and the scanner starts searching for frequencies with intermittent activity. During this search continuous signals are filtered out and frequencies with intermittent activity are selected in steps of 25 kHz.

The number of times the entire band is searched can be set from the terminal with the 'P' command. Each scan pass takes 7 minutes, so the search time with P = 2 is 14 minutes. A good option is to set P to 6 (search time 42 minutes). Best results are obtained with P = 15, but this means you have to wait nearly two hours.

At the end of the search the frequencies that have been found are programmed in the memory, after which LED D3 goes out and the receiver must be restarted. N frequencies are programmed, starting at memory location M (see the description of the 'M' and 'N' commands). It's important to adjust the squelch potentiometer to a good setting (with the aid of LED D4) before starting the search. The resulting frequencies are not scanned until the scanner is restarted. They are not stored in EEPROM until LED D3 goes out, so nothing happens if the scanner is switched off prematurely.

programmed microcontroller and a PCB are available for building the scanner.

(100696-I)

Internet Links

[1] www.elektor.com/bob

[2] www.elektor-magazine.com/100696
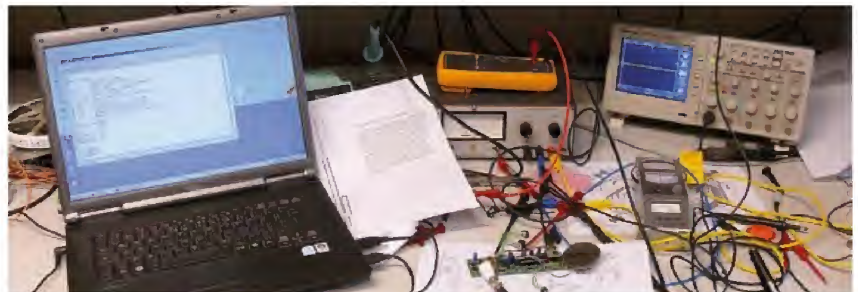
Humidity sensor
Handy for in the bathroom

By Leo van der Linde (New Zealand)

COMPONENT LIST

Resistors
R1,R2,R3 = 220kΩ
R4 = 1kΩ
P1 = 100kΩ trimpot, horizontal

Capacitors
C1,C2,C4,C5 = 100nF
C3,C7 = 1000µF 16V radial
C6 = 3.3nF

Semiconductors
B1 = B40C1500 bridge rectifier
IC1 = 78L05
IC2 = NE556
IC3 = S101S06V (Sharp) solid-state relay

Miscellaneous
Tr1 = AC power transformer, PCB mount, sec. 2 x 6V, 350mW (Block type AVB 0,35/2/6)
K1, K2 = 2-way PCB terminal block, lead pitch 7.5mm
K3 = 2-pin pinheader for sensor connection
Sensor: Hygrosens type SHS A2 (Octopart.com)
PCB # 120463-1 (www.elektor-magazine.com/120463)

This circuit was devised by the author to automatically turn on an extraction fan in the bathroom when the humidity becomes too high. The detector circuit consists mainly of a dual CMOS timer, a resistive humidity sensor and a solid-state relay.

Timer IC2A is configured as a 1-kHz oscillator with the aid of R1 and C6. The output signal goes via a resistor (R3) to the humidity sensor, which is connected to K3, and to the trigger input of the second timer, IC2B. In series with the sensor are a capacitor (to prevent polarization of the sensor as a consequence of the DC offset voltage) and a trim-pot (to set the sensitivity). As a result of the exponential resistance characteristic (< 20 kΩ at 75 % RH, < 100 kΩ at 93 % RH and < 150 kΩ at 95 % RH), the resistance of the sensor will eventually become sufficiently high as the humidity increases, so that the output signal of IC2A will trigger the second timer IC2B (configured as a retriggerable monostable multivibrator). This second timer will then switch on the extraction fan via solid-state relay IC3. As long as the humidity remains high the fan will remain in operation. When the humidity drops there will be a point at which IC2B no longer receives the trigger signal. The timer will then continue to run the fan for another 4 minutes and then switch it off.

When the circuit is powered up for the first time the fan will immediately turn on for 4 minutes. After that the fan will only turn on when the humidity in the bathroom is too high.

Keeping safety in mind, the power supply for the circuit consists of an AC line transformer followed by a bridge rectifier, electrolytic filter capacitor and regulator. This method prevents the sensor (which has to be in the bathroom) from having a dangerously high voltage on it.

A small printed circuit board has been designed for the circuit, which is quite easy to assemble (since only leaded components have been used). Make sure that you fit the wire link for the selection of the AC grid voltage in the appropriate place.

Because of the high humidity in the bathroom it is essential that the circuit is mounted in a drip-waterproof housing. If you are lucky, the fan housing may be large enough to accommodate this board as well. However, the sensor itself has to be placed in a location where it will be in contact with the damp air in the bathroom (if necessary let it protrude from the housing and make the feed-through waterproof using, for example, silicon sealant).

Connecting the circuit doesn't really require any further explanation: The AC voltage is connected to K1 (which was originally connected directly to the fan), the fan is now connected to K2. Using P1, find the appropriate setting so that the fan switches on when the relative humidity is above about 90 %. This is easily done 'by feel'.

(120463-I)

Aviation Band Antenna
Ideal for the 108–137 MHz scanner

By Gert Baars (Netherlands)

This end-fed or "Zepp" or "J-Pole" antenna provides excellent results in combination with the scanner for the aviation band (108–137 MHz) described elsewhere in this edition. It can be made from standard coax cable, such as RG58 or the thinner RG174, but also thicker types like RG8 or RG213.

The antenna consists of a half-wave whip and two pieces of coax that transform its impedance

to 50 Ω. To make the antenna, start by cutting two pieces of coax cable — one with a length of 12 cm (4.72 inch) and the other with a length of 36 cm (14.17 inch), plus a centimeter (.49 inch) for the connections. Short out one end of the shorter piece by soldering the shield and the inner conductor together. Connect the other end in parallel with the longer piece and a length of cable for connection to the scanner input. You can also use a connector for this.

At the far end of the longer piece of coax, solder the inner conductor to a whip antenna with a length of 117 cm (46.1 inch). This can be made from solid 1-mm wire (#18 AWG) or similar material, and it can be hung up somewhere indoors.

For use outdoors, you can slide the entire arrangement into a length of PVC pipe or a fishing rod protector.

(120678-I)
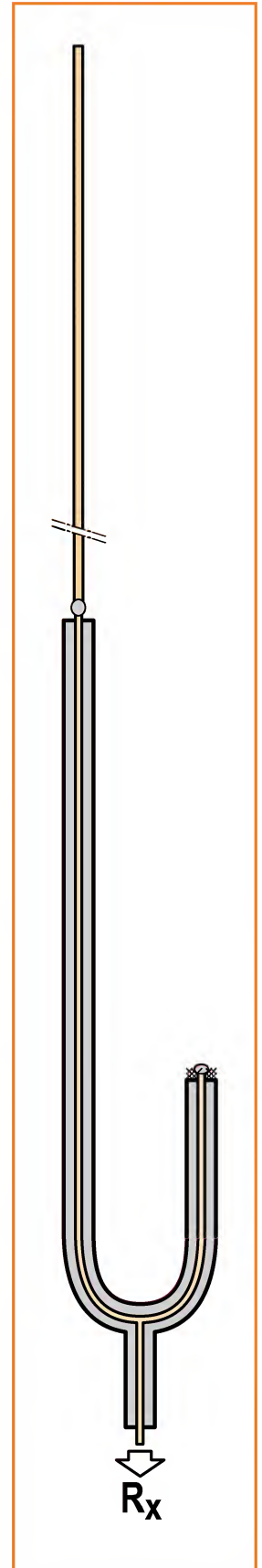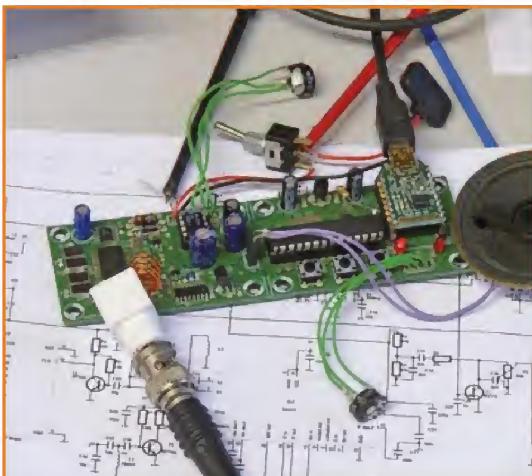
Taming the Beast (2)
How to build an LED blinker
with 250 Kgates
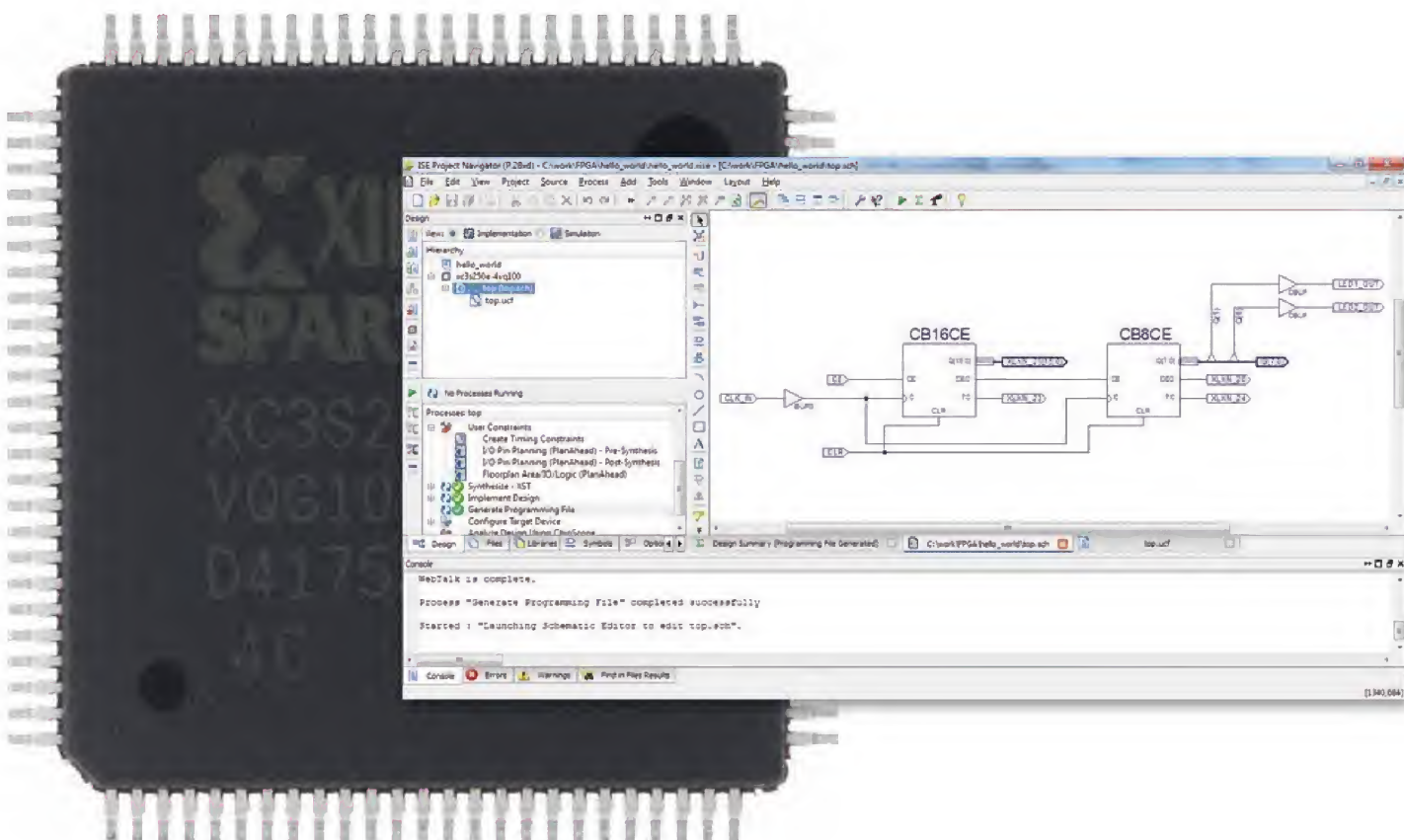
Text:
Clemens Valens
(Elektor Lab)

Design:
Raymond Vermeulen
(Elektor Labs)

In the previous article in this series we described the basic features of FPGAs and what they can be used for. We also presented a simple development board that you can use to gain experience with FPGAs. FPGAs are complex devices, and the tools for programming them are just as complex, so our aim is to help you get started one step at a time. As you will see, the first steps are relatively easy, but if you want to delve deeper into this subject you will have to put more effort into it.

First step: installing the software

So that you can get started right away, in the previous article we recommended that you request the free DVD with the necessary tools from Xilinx, and we hope that you now have it on hand. One of the advantages of this DVD is that it comes in two versions: one for Windows and the other for Linux.

You can also download everything, but it appears that the passion of FPGA manufacturers for packing as many gates as possible into their compo-

nents also extends to stuffing as many bytes as possible into their tools; in any case the download is more than 6 GB. If you opt for the download route, make sure that Xilinx Download Manager works properly with your browser (**Figure 1**). I had a problem with Firefox, probably due to some of the configuration settings, but it worked okay with Internet Explorer. The advantage of this download manager is that it can resume the download if it gets interrupted, which can save a lot of time. A drawback (at least in my case) is the low speed.

I don't have a Linux PC, so in the rest of this article I'll be working with the Windows version, but I assume that the installation procedure for Linux users is not much different. Insert the DVD in the computer or extract the files from the downloaded TAR file, and then launch the program *xsetup.exe* (if it does not start automatically). Click *Next*, accept the two license agreements, and then select *ISE WebPack* (**Figure 2**). Click *Next* again and leave all the options as they are (to be on the safe side). In theory you only need to tick *Acquire or Manage a License Key,* but who knows what else you might want to do later after you understand things better. Surprisingly enough, this has no effect at all on the size of the installation (over 12 GB). Click *Next*, choose a location for installing everything, click *Next* again, browse through the summary and then click *Install*. Depending on the PC you're using, you have time to do something else now – in my case the installation took about 45 minutes. Now you have to request a license. In the *License Configuration Manager* window (**Figure 3**), select *Get Free ISE WebPack License* and then click *Next*. Now you will see a window with the data to be used for the license. Click *Connect Now*. This takes you to the Xilinx website, where you first have to create an account if you do not already have one. If you have an account, you can proceed directly. If you have to create an account, you will end up a long way away from the license page you are actually interested in. The easiest way to get back to it is to click the *Acquire a License* tab in the *License Configuration Manager* window. This puts you back to where you were in Figure 3, from where you can proceed as previously described.

In order to reach the license request page you have to fill in several forms, which you should
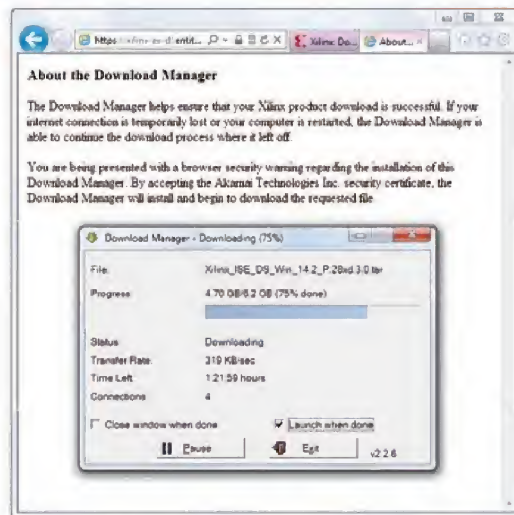
Figure 1.
Xilinx Download Manager window. If you don't see this, it isn't working.
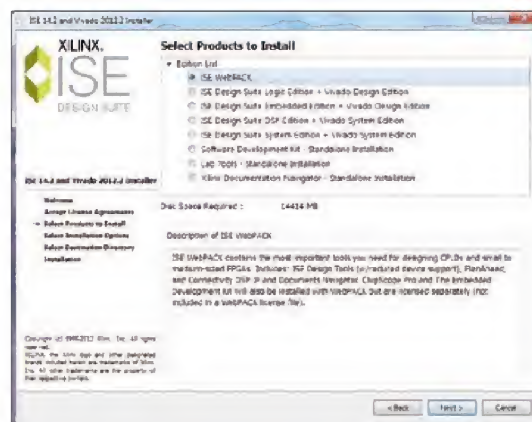
Figure 2.
ISE Installer, which is what you use to install ISE WebPack.
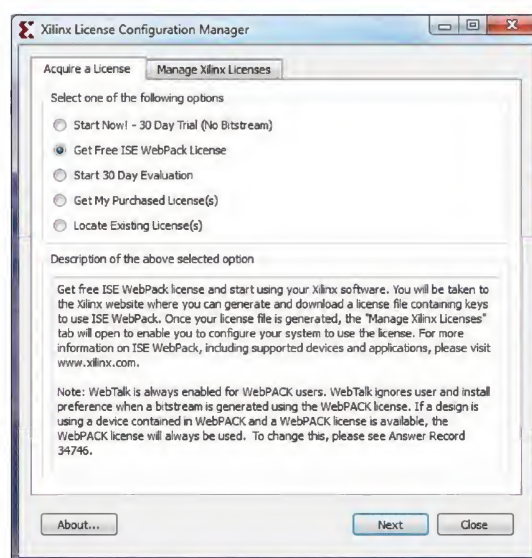
Figure 3.
You need a license in order to use ISE WebPack.
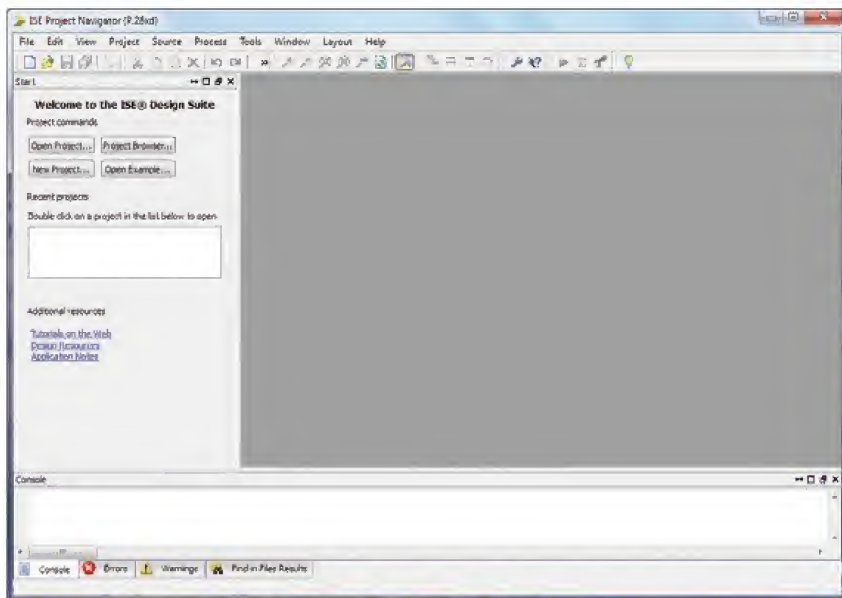
Figure 4.
ISE Project Manager is completely empty at this point

complete as you see fit. On the license request page, select *ISE Design Suite: WebPack License* and click the *Generate Node-Locked License* button. This brings up a summary with a *Next* button, which is followed by yet another summary with a *Next* button, and finally a congratulations message telling you that the license file has been sent to the e-mail address you entered when you created your account. Put this file somewhere on your PC in a place where you can find it again, and then go back to *License Configuration Manager*. This time click the *Manage Xilinx Licenses* tab and then click the *Copy License...* button. Navigate to the license file and click *Open*. This should result in the following message: "*License installation was successful.*" If it does, you can exit the license manager by clicking *Close*. You can also close the installation window by clicking *Finish*. You'll probably agree with me that this isn't the sort of installation you whiz through between other jobs.

Second step: getting started

Once you have installed the ISE tools and accepted the license, you can start doing some real work. You should see a new item – *Xilinx Design Tools* – in your *All Programs* list. Select this item, then select *ISE Design Suite 14.2* (the version number may be different), then select *ISE Design Tools*, and finally launch *Project Navigator*. Two versions are available on my system: a 32-bit version and a 64-bit version. Select the version that matches your PC. Now you will see the window shown in **Figure 4**.

Click *New Project...* to open the *New Project Wizard*. Fill in the form as shown in **Figure 5**, but with a name that makes sense to you and a location that works on your computer. You can enter a description in the *Description* field — a good habit to learn. Here we describe how to develop your first FPGA application in the form of a schematic diagram, since most of us are familiar with this form. Native FPGA programming with a language such as VHDL or Verilog can come later. Select *Schematic* as the *Top-level source type*, since that's what you will use. FPGA projects generally have a hierarchical structure because they are often rather large. This means that they have several levels, with more and more detail as you go down to lower levels. The highest level — the top level — is in fact simply a block diagram. Each block is broken down into bite-sized blocks at successively lower levels, until the function is finally defined in detail. For your first project — which will cause a LED on the FPGA development board to blink — it is not necessary to work with multiple levels, so for now you can do everything at the top level.

Click *Next* to allow the wizard to present its second page (**Figure 6**). You can leave the default values in most of the fields, but not in the *Family*, *Device*, *Package* and *Speed* fields. They must match the device on the board, which in this case is a device from the Spartan 3E family, device type XC3S250E, in a VQ100 package. For *Speed* select *-4* for the standard version (*-5* is the fast version). This is also marked on the IC; you will see "4C" at the bottom if you look closely. Now click *Next* to view the project summary. This is only for your information, since you can't change anything at this point, so you can simply click *Finish* right away.

Now you are facing an empty project. Take a bit of time to look around before you open the *New Source Wizard* (**Figure 7**) by selecting *Project → New Source...*. Select *Schematic* as the *Source Type*, enter *top* as the file name, and check that *Add to project* is ticked. Then click *Next* again. This brings up a summary window that you close by clicking *Finish*.

After a bit of flash and flutter the ISE layout suddenly changes to an empty window where you can draw a schematic. There are also a few changes to the left of the "drawing sheet", and

now you can see a list of options there. Below this list there are several tabs – many more than you can see at once. Click the arrow to the right of the *Options* tab a couple of times to view all the other tabs. The *Design* tab is what you were looking at before you launched the *New Source Wizard*. As you can see here, the schematic diagram *top* has been added to the project.

Third step: drawing a schematic

Our objective here is make a LED on the FPGA board blink visibly. In this case "visibly" means a blinking rate less than 10 Hz or so. The FPGA is clocked at 8 MHz by the microcontroller. You can generate a usable frequency from this by dividing the clock signal by 2 several times, which is easy to do with flip-flops. If you divide 8 MHz by 2 twenty-two times, the resulting frequency is a bit less than 2 Hz (8 MHz / 2^{22} = 1.9 Hz), which is just what you need.

You could draw 22 flip-flops on your schematic, but you can also use a counter instead. To do this, look on the *Symbols* tab next to the *Options* tab in the bottom left corner of the ISE window. Then select *Counter* in the *Categories* list. The *Symbols* list shows you all the available counters. It does not include a 22-bit counter, so you will have to connect two smaller counters in series. The first counter in the list I see is called *cb16ce*, which sounds like a 16-bitter, so let's start with that.

Placing symbols

Select the *cb16ce* counter by clicking it and then moving the mouse to the schematic pane, which will cause a symbol outline to appear beneath the mouse cursor. Click somewhere in the schematic pane to place the counter. Now you can see that it is indeed a 16-bit counter, with a number of extra inputs and outputs. To find out what all the pins are for, right-click the symbol. In the pop-up menu, select *Symbol* and then *Symbol Info*. This opens *ISE Help* on the page containing a description of symbol concerned. For example, you can see that in order to enable counting, the CLR pin must be low and the CE pin must be high. You can also see that the CEO pin can be used to clock a subsequent counter. This means that you do not need the Q9–Q15 outputs. For the second stage of the counter, place a *cb8ce*. From the previous article, you should know that I/O blocks are located between the logic blocks of the FPGA and the outside world. Every sig-

nal that enters or leaves an FPGA pin must pass through an I/O block. This applies to the LED signal, the clock signal, and the CE and CLR signals, which we want to be able to control via pins. ISE has specific symbols for this purpose, which are
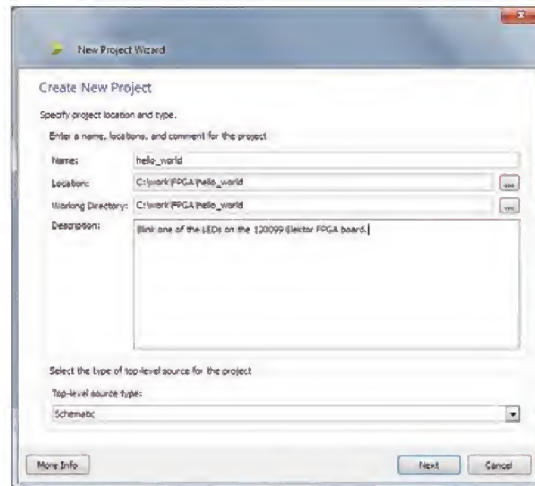
Figure 5.
Use the New Project Wizard to create new projects.
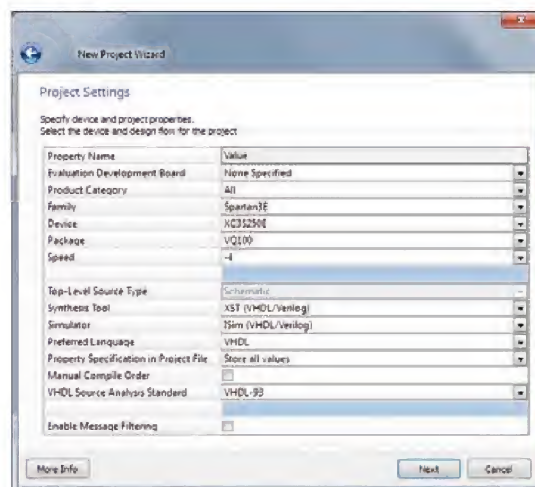
Figure 6.
It's important to select the right FPGA.
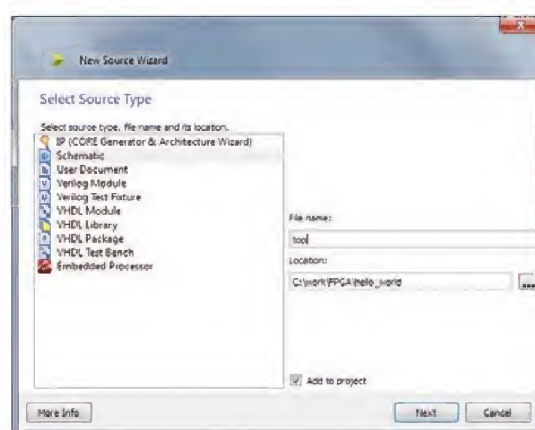
Figure 7.
A wizard helps you add files to your project.

Microcontroller firmware: the basics
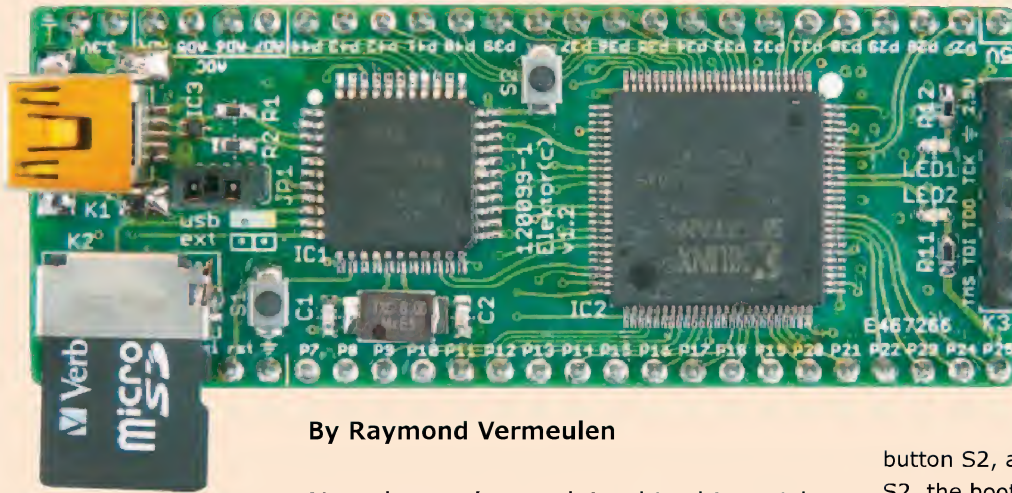
By Raymond Vermeulen

Now that we've explained in this article how to generate a .bin file in ISE Design Suite, a good many readers are probably wondering how the file gets loaded into the FPGA in the right way. This task is handled by the microcontroller on the board. Here I want to describe the firmware developed in the Elektor lab for this purpose. For anyone who is interested, the source code and hex code are available on our website [1]. The code is fully open source, and we encourage our more experienced readers to add more functions to this code.

The underlying design philosophy for the development of this code was that you shouldn't re-invent the wheel every time. Open-source code, and in some cases even complete libraries, is available for many functions, and we used this code as appropriate. In particular I would like to thank Dean Camera for the LUFA library [3] and ChaN for (Petit) FatFs [4]. Without this code, the project would have taken a lot longer and would probably have had fewer features. I used version 120219 of the LUFA library and version R0.02a of Petit FatFs.

An ISP is not necessary for updating the firmware in the microcontroller. A DSP boot loader is integrated in the device (this is part of the LUFA project). The firmware can be updated very easily from Windows by using the Atmel Flip software [5].

The boot loader is normally skipped when the board starts up, but if you press and hold the reset button (S1) followed by button S2, and then release S1 before releasing S2, the boot loader is launched. The associated driver is included in the installation file for Atmel Flip.

Now let's get back to the actual application. After start-up the FPGA is first programmed, and then the USB-related tasks are executed. I chose this sequence because LUFA uses interrupts to handle USB transactions. It seemed like a good idea to complete very time-critical tasks before enabling the interrupts. This also allowed me to keep the two main functions separated. Another relevant factor is that the microSD card cannot be mounted on the PC and locally at the same time.

In the first phase the hardware is initialized. The necessary I/O settings are made, such as configuring the m{2:0} pins of the FPGA in SerialSlave mode. Hswap is also set high to disable the pull-ups of the I/O pins during configuration. The microSD card is queried, identified and prepared for use (various versions require non-standard communication). The partition on the microSD card can be mounted locally using Petit FatFs functions. If the file *config. bin* is present on the card, it is opened. The *Prog_b* line is held low for 1 ms to reset the FPGA, and then the *Init_B* line is set high. Now the data transfer can start. Data blocks are read in from the microSD card and written to the FPGA via the

USART bus, which is driven using functions from the LUFA library. While these data blocks are being transferred, the *Init_B* line is checked after each transmitted byte to see whether it has gone low, which would indicate an error. After the entire file has been transferred, the *Done* line is checked to see whether it has gone high, which indicates that the configuration was completed successfully. After this the microSD card is dismounted, the SPI and USART peripherals are disabled, and the pins that were used are restored to the high-impedance (Hi-Z) state.
Petit FatFs is used for local handling of FAT32 and FAT16 without the aid of an OS. This is located in *FPGA_board_Config_v1_0\local_fat* in the Zip file that can be downloaded from [1]. In *pff.h* define statements are used to provide the option of supporting FAT32. Only read access is enabled, since writing to the microSD card is not necessary in this phase. *MMC.c* is part of an AVR demonstration application available at [3]. The previous low-level functions have been largely replaced by LUFA library functions.

In the second phase an endless loop is started to handle USB communication, using a virtual COM port and the microSD card as the mass storage device. For this purpose the hardware is again initialized, which means that the microSD is again queried, identified and prepared for SPI communication. The results from error handing in the first phase are used to select a string that controls the virtual COM port for communication with the PC. The microSD card is now handled by the OS as a USB stick, which means that SCSI commands are sent to the card over the USB bus and that the file system is not running locally any more, but instead on the PC. Code for using a microSD card in this way is not included

in the standard LUFA package, but I found some on the Internet. I modified it so that it can work with LUFA version 120219. A certain amount of code that is now superfluous is still present in the *FPGA_board_Config.h* file; it can be deleted at a later date.

The LUFA library is primarily used to handle all USB matters, but there are also a lot of low-level functions in the library for driving various peripheral devices. The LUFA package includes hardware profiles for communicating with specific boards. I could have added a profile for our development board, but in the heat of the battle I sacrificed compatibility in the interest of faster and easier programming. Demo projects are also included in the LUFA package. The present firmware is based on the VirtualMassStorage demo. For more information on the detailed operation of LUFA, the accompanying documentation can be downloaded [3].

For debugging purposes, the code uses a virtual COM port that sends error messages to the PC. This proved to be very helpful during development. Anyone who is interested can use it for the same purpose, but with some modifications it can also be used for communication in the opposite direction.

Tip:
Define statements can be used for simple, function-like notation. This makes for tidy code without incurring function overhead. I used this to make it easy to set or read some pins while keeping the function code readable.
For example, the following defines are present in *FPGA_board_Config.h*:

```
#define Release_FPGA_reset()     DDRB&=~(1<<DDB4)
#define Init_B_status()          ((PINC >> PC6) & 0x01)
```

They are used in a function in *FPGA_board_Config.c*, such as:

```
while (!Init_B_status()) {;;}
```

or:

```
Release_FPGA_reset();    // release the Prog_b line
[/code]
```

•Projects

Figure 8.
The schematic diagram of
the LED blinker.
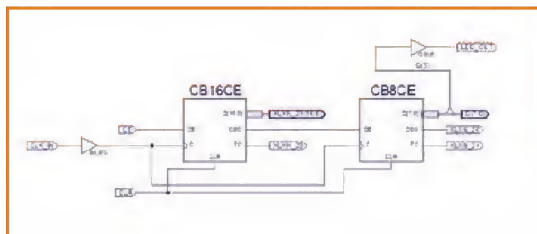
located in the *IO* category. There are some subtleties involved in this, so pay good attention here. Clock signals are special, so there is a special symbol for creating a clock input. It is called *ibufg*. This buffer cause the clock signal to be connected directly to one of the dedicated clock lines in the FPGA. The other inputs can also be connected via buffers by using the *ibuf* symbol. However, if you take a close look at the help info for this symbol you will see that it is better not to place these symbols yourself, but instead allow ISE to place them as much as possible. What you do have to place yourself is the I/O markers, which act as labels. Accordingly, in your simple design you should place I/O markers but no *ibuf* symbols. However, outputs must be explicitly fed out via buffers. There are various symbols that can be used to specify the output type. For the LED, you can use a simple *obuf*.

Accordingly, you should now place an *obuf* in the vicinity of the Q output of the *cb8ce* counter and an *ibufg* next to the clock input of the *cb16ce* counter.

Routing lines

Now you can start making connections by clicking the *Add Wire* button, which is located on the toolbar between the schematic diagram and the *Options* tab. If you hover the mouse cursor above a button, you will see a description of the button function. After you click the *Add Wire* button, the mouse cursor changes to a crosshair and you can click the pins in the schematic diagram. The default mode is *Autorouter* (see the *Options* tab), which means that you only have to click the start and end points of a connection and ISE will look after routing the line in a more or less tidy manner. You can use the mouse to drag line segments in order to make the drawing a bit nicer. Using this method, connect C to C and CLR to CLR on both counters. Connect CEO of *cb16ce* to CE of *cb8ce*. Connect the output of *ibufg* to the C input of *cb16ce*. Leave the input of *obuf* unconnected; this will be taken care of later.

Setting markers

Next you have to add I/O markers with the *Add I/O Marker* tool. If the option *Add an automatic marker* is enabled, you can click a pin directly to add a marker in some cases. If this does not work, first use *Add Wire* to draw a line segment or a bus segment. Double-click to end a segment at an arbitrary location. Be sure to draw a bus segment at the Q[7:0] output of *cb8ce*, since you will need this right away.

I/O markers are assigned names automatically. You can use the *Add Net Name* tool to change these names. First enter the name of the net (*Options* tab) before clicking the net you want to rename. Using this method, rename every net that is routed to a pin (CLR, CE, CLK_IN and LED_OUT). Rename the bus at the Q[7:0] output of *cb8ce* from *cb8ce* to *Q(7:0)* (note that if you use square brackets, ISE will change them to round brackets). Q(7:0) designates a bus with signals Q0 to Q7.

Connecting bus lines

The only thing left to do now is to connect the input of *obuf* to the Q5 output of counter *cb8ce* (so that the counter chain will divide by 2^{22}). To do this in a manner that ISE finds acceptable, click the *Add Bus Tap* button and then select *Bottom* (or something else, as appropriate) under *Options* to get the right orientation. Now you can place a triangular symbol on the schematic with the base on the Q(7:0) bus segment. Then connect the tip of the triangle to the input of the *obuf*, and name this connection *Q(5)*. Your schematic should look something like the one shown in **Figure 8**.

Before continuing with assigning markers to FPGA pins, it is advisable to check whether there are any problems with the design. Normally speaking you should simulate the design at this point, but the design is so simple (and this article is already so long) that you can skip this for now. However, you should check that it is possible to synthesize the design, which means asking ISE to translate the schematic into standard gates and flip-flops available in its libraries. For the programmers among you, this is comparable to compiling a program.

Synthesizing the design

To synthesize your design, click *Synthesize – XST* on the *Design* tab. You will see all sorts of messages in the *Console* window, and a rotat-
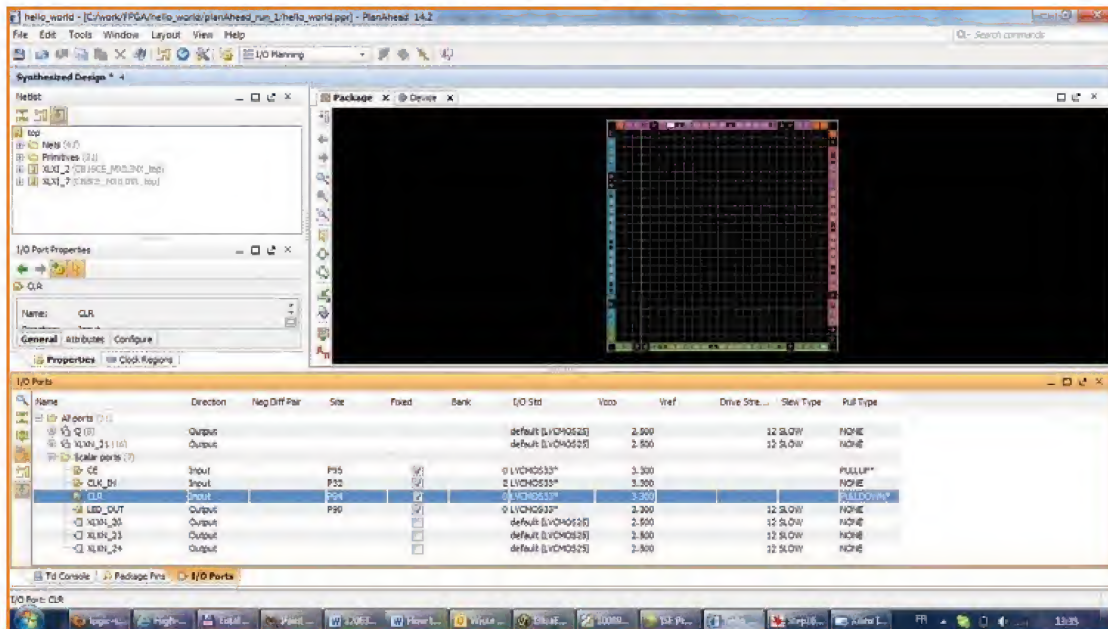
Figure 9.
This nice graphic tool lets you assign FPGA pins to signals. Alternatively, you can edit the UCF file in a text editor.

ing icon indicating that something is happening. If everything works out okay, a big check-mark will appear to the left of *Synthesize – XST*. If you look in the console window, you will see that the design uses 24 flip-flops.

Assigning pins
Now you have to assign pins to the markers. You do this with the *PlanAhead* tool, which you can launch from the *Design* tab. For this you have to open the *User Constraints* drop-down menu and double-click *I/O Pin Planning (PlanAhead) Post-Synthesis*, since you just synthesized the design. ISE asks whether you want to add a *User Constraint File* (UCF) to the project, which you in fact do want (trust me), so you should click *OK*. It may take a while for *PlanAhead* to complete its task, depending on your PC, because it has a lot to do.

The markers in your schematic diagram are shown on the *I/O Ports* tab of PlanAhead (at the bottom). If you open the *Scalar ports* drop-down list, you will see the markers that need to be connected to pins. At this point it's helpful to refer to Figures 7 and 8 in the previous article, so you know which pins you can use. The LEDs are connected to pins 90 and 91. Look in the row *LED_OUT* and the column *Site*. In the drop-down list that appears, select P90 (or P91 if you prefer). Then click the *I/O Std* column and select *LVC-MOS33* (the only standard that can be used on

the development board). Now you're done here; the rest will be filled in by the tool. Next comes the *CLK_IN* pin. The clock input for the FPGA on the development board is normally pin 32, so you should select *P32* in the *Site* column. Here again, set the I/O standard to LVCMOS33. Now you're left with the *CE* and *CLR* lines. You're free to make your own choice here, as long as the pins are routed to connector K4 or K5. I chose P94 for *CLR* and P95 for *CE*. The I/O standard is again LVCMOS33. To ensure that these inputs do not block the operation of the circuit, you should also enable pull-up and pull-down resistors on these pins. In the *Pull Type* column, enter *PULLUP* for *CE* and *PULLDOWN* for *CLR* (**Figure 9**). Save your configuration by clicking the *Save Constraints* button, and then go back to the ISE *Design* tab.

Implementing the design
The next step is to implement the design, which means translating it into logic blocks and so on. Double-click *Implement Design* and wait until a green check-mark appears. If there are any errors or warnings, you can view them in the *Errors* and/or *Warnings* tab(s) at the bottom. If you did everything exactly as described, you should not see any errors or warnings.

Generating the bitstream file
You're almost done. What you have to do now is to generate a file that can be used to program the FPGA. This is called the bitstream file, and
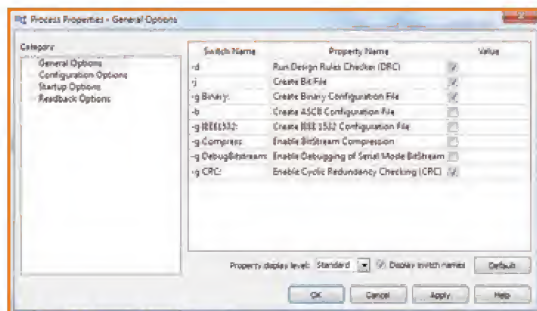
Figure 10.
Bitstream options (page 1).
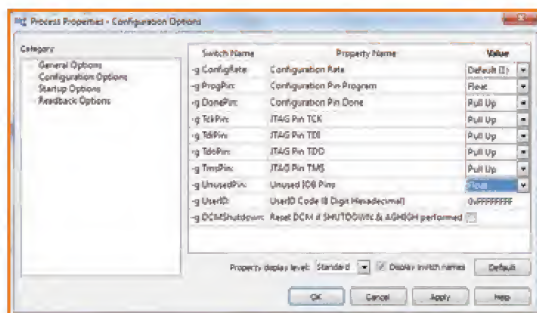
Figure 11.
Bitstream options (page 2).
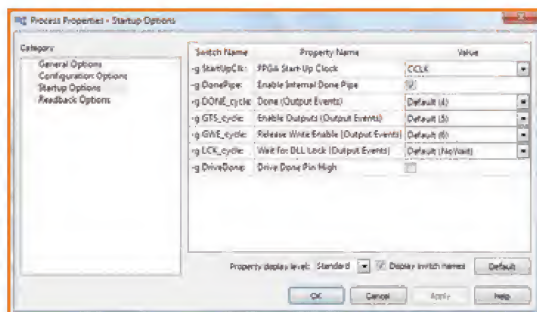
Figure 12.
Bitstream options (page 3).
No changes are necessary
on the fourth page.

Programming the FPGA

Plug the microSD card into the FPGA board and connect the board to a free USB port on your PC. The PC should recognize it as an external mass storage device. Copy the .bin file you just generated to this external storage device. Then rename the file *config.bin*. Remove the external storage device (use the Windows safe removal procedure) and then press button S1 (close to the microSD card on the board). If everything's as it should be, one of the LEDs should start blinking.

Homework

Your homework assignment is to connect the other LED on the FPGA board to the circuit so that it also blinks, but in a different way.

(120630-I)

Web links

[1] www.elektor-magazine.com/120630

[2] www.xilinx.com/support/download/index.htm

[3] www.fourwalledcubicle.com/LUFA.php

[4] http://elm-chan.org/fsw/ff/00index_p.html

[5] www.atmel.com/tools/FLIP.aspx

there are many different types. To ensure that you generate the right type, right-click *Generate Programming File* on the *Design* tab and select *Process Properties* in the pop-up menu. Select *General Options* in the *Category* list and tick *Create Binary Configuration File* (**Figure 10**). Select the next category, *Configuration Options*, and set the values of *Configuration Pin Program* and *Unused IOB Pins* to *Float* (**Figure 11**). Tick the *Drive Done Pin High* option in the *Startup Options* category (**Figure 12**). You can leave the *Readback Options* settings unchanged. Click *OK* to close the *Process Properties* window, and then double-click *Generate Programming File* to generate the bitstream file. After a while a green checkmark should appear here as well. You should now see the file *top.bin* in the project folder (see the ISE title bar).

Embedded Linux Made Easy (7)

by
Benedikt Sauter [1]

I²C, serial ports and RS485

In the previous installment in this series we called on our readers to tell us what they would like to see in this final part. Many thanks to the well over one hundred readers who responded to the call! By popular demand, therefore, we will look more closely at how a real application is developed. The I²C and serial interfaces, which are often needed in real-world projects, provide the other main focus of this article.
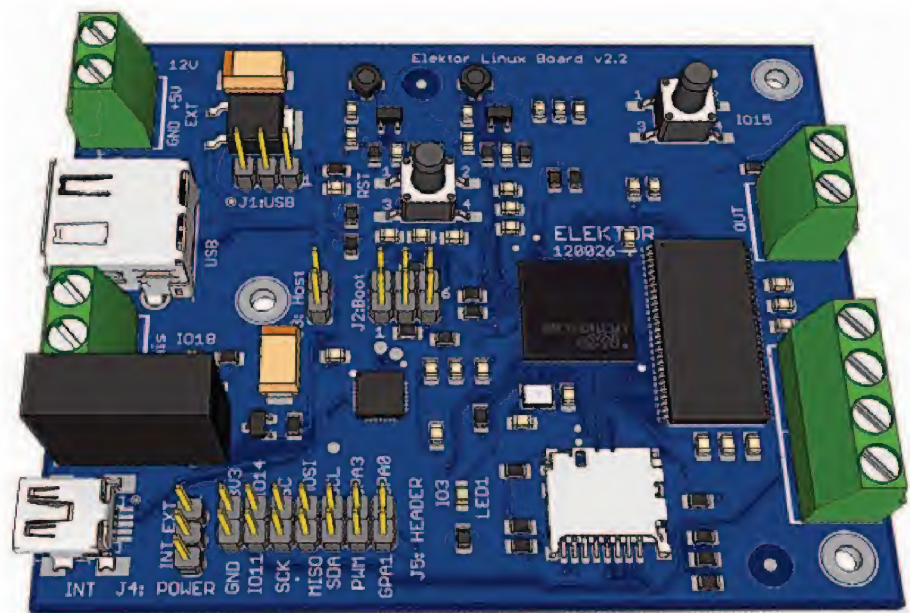
3D Model of the Elektor Linux board created using the EagleUp tool [14].

Open source software's lifeblood is open development in the community. Individual project websites therefore form a source of updates, patches and extensions, and they often also provide useful hints and tips and even user forums.

The Elektor Linux board is based on the Gnublin project [2], which was launched in 2009 by the author in association with the Augsburg University of Applied Sciences, Germany, as a platform for embedded Linux development. Since then several variations on the board have been produced, as well as compatible extension circuits to drive stepper motors and interface to the CAN bus.

And, of course, the software has been extended in many different ways, including updates to the kernel and to the toolchain, including the C/C++ compiler. Let's install these first!

Updates

In the third part of this series [3] we installed a C/C++ compiler on the PC to avoid having to compile programs directly on the *Elektor* Linux board. We can now update to the most recent version of this compiler, along with the other components in the toolchain.

The easiest way to do this is to open a terminal

window on the Ubuntu Linux machine using the control-alt-T key combination. Type the following command into the console:

```
wget http://gnublin.org/downloads/eldk-
eglibc-i686-arm-toolchain-qte-5.2.1.tar.
bz2
```

When the archive has finished downloading, we can unpack it on top of the local file system:

```
sudo tar xjf eldk-eglibc-i686-arm-
toolchain-qte-5.2.1.tar.bz2 -C /
```

Before we can use the new compiler we have to write a small script to automatically set the environment variables to point to the new installation's directories. Create a file called 'set.sh' in the directory '/opt/eldk-5.2.1':

```
sudo gedit /opt/eldk-5.2.1/set.sh
```

Figure 1 shows how the new directories are added to the PATH environment variable. Save the script file you have created and run it from the console as follows (watch the 'period-space-forward slash' syntax at the start):

```
. /opt/eldk-5.2.1/set.sh
```

From now on this command must be re-entered every time you want to compile a program for the *Elektor* Linux board on the development PC. The compiler in the new toolchain can be tested using the following command:

```
arm-linux-gnueabi-gcc -v
```

C/C++ development environment

Many readers asked about the possibility of developing C/C++ code for the board using a user-friendly integrated development environment (IDE) such as Eclipse. We will therefore briefly look now at an easy way to develop C/C++ programs on the development machine and then test them immediately on the board.

You will need the following:

- console access to the Linux board (over its serial port or over the network);
- a network file system (to allow access to files from the development PC);

- and, if you wish, the development environment itself.

Since devices appear just like files within the Linux directory hierarchy, it is natural to want to have the remote storage devices appear in the directory structure of the development computer just like any other directory or subdirectory. Under Linux there are (of course) many options open to achieve this, including NFS, FTP and Samba.

In the previous installment of this series [4] we looked at remote maintenance of the Linux board using SSH. SSH is a client-server system that allows access to a console on the board over a network connection. SSH is similar to Telnet, but provides a higher level of security by encrypting the connection between server and client. It also provides a mechanism for transferring files between the server and the client, using (on the client side) a program called 'scp'.

```
P1=/opt/eldk-5.2.1/armv5te/sysroots/i686-eldk-linux/usr/bin/armv5te-linux-gnueabi/
P2=/opt/eldk-5.2.1/armv5te/sysroots/i686-eldk-linux/bin/armv5te-linux-gnueabi/
export ARCH=arm
export CROSS_COMPILE=arm-linux-gnueabi-
export PATH=$P1:$P2:$PATH
```

Figure 1.
Environment variables for the new toolchain.

If the Linux board has the IP address 192.168.0.190, then we can copy a file directly from the development PC into the Linux board's file system using the following command:

```
scp test.c root@192.168.0.190:/root
```

To copy an entire directory, use a command such as:

```
scp -r myproject root@192.168.0.190:/root
```

The first argument to the command is the file or directory to be copied. If a whole directory is to be copied, it must be preceded by the '-r' option, which causes a 'recursive' copy of all files and directories found under the specified directory. The second argument consists of the user name, followed by an '@' character and the IP address, and then a colon and the destination path on the Linux board for the copy operation.
This gives us a convenient way to copy compiled

code to the board whenever we wish. However, we can make things even easier!

The slickest approach is to install the Linux program 'sshfs'. This is done by typing the following command on the development PC:

```
sudo apt-get install sshfs
```

The tool allows us to access the entire file system of the Linux board live from the development PC, completely avoiding the need to copy files across to the SD card by hand. All that is needed (as indeed for SSH and scp) is of course a working network connection.

We first have to create a so-called 'mount point' within the development PC's file system. This is the point in the file system where the files that are actually on the remote system will appear. The mount point is normally an empty directory, which we can create either using the 'mkdir' command at the console or using a graphical file manager.

First switch to your home directory:

```
cd ~
```

and then create the mount point:

```
mkdir elektor-linux-board
```

The next command will make the whole file system of the Linux board available on the development PC:

```
sshfs root@192.168.0.190:/
elektor-linux-board
```

When the command is entered you will be prompted one last time for the password. When this has been entered successfully, you will be able to navigate around the remote file system using commands like 'cd' and 'ls' as usual. However, as you wander around you may occasionally be greeted by the message 'Permission denied'. This is because the system programs on the Elektor Linux board are owned by the user 'root' and are not readable by ordinary users; as far as the remote file system is concerned your identity (or 'uid') is that which you have on the development machine. In order to be able to roam freely around the remote file system you must adopt the

root identity on the development machine. The simplest way to do this is to start a new console with root privileges:

```
sudo /bin/bash
```

In this console you will be free to examine, create and delete any file you choose.

Now, in order to write programs for the board on the development PC, we simply need to put the project directory in the mounted file system. You can then use Eclipse or other environment as normal: see [5] for more information.

It is preferable to create a dedicated user on the *Elektor* Linux board before commencing a new project. This can be done as follows:

```
adduser elektor
```

You can then mount this user's home directory in the development PC's file system:

```
sshfs elektor@192.168.0.190:/home/elektor
elektor-linux-board
```

If you are the only user on both the development PC and the board, in each case your uid will be 1000. This means that you will not encounter any problems with file permissions between the two machines. In any case, as a fall-back, it is always possible to run programs as the root user.

Kernel update

The kernel provides the interface between applications and hardware. If you want to use additional or new hardware, it will sometimes be necessary to download a new version of the kernel and compile it manually.

The Gnublin project's kernel archive is the first port of call for *Elektor* Linux board users. The kernel has been extended since the first article in this series was published, adding support for CAN, SPI and I^2C devices.

The source code is kept centrally in a version control system, which provides a place where developers can easily make changes themselves and track changes made by others. The 'git' version control software is used to manage the archive. To access a git repository the git package is needed: on the development machine type the following:

```
sudo apt-get install git-core
```

You can now 'clone' the archive from the repository into any directory you choose:

```
git clone http://code.google.com/p/
gnublin-develop-kernel/
```

This operation may take some time. When the downloading has finished and the prompt returns, you are ready to 'build' the most recent stable kernel for the board. Change to the directory for version 2.6.33 of the kernel:

```
cd linux-2.6.33-lpc313x
```

(Alternatively, if you are feeling brave, you can try out the experimental version 3.3.0.) In the directory create a suitable default configuration:

```
make elektor_defconfig
```

If you wish to make manual changes to the kernel configuration, you can do this as before with:

```
make menuconfig
```

Finally you can build the kernel (not forgetting to run 'set.sh' first):

```
make zImage
```

Assuming the kernel builds without errors, it can now be copied onto the Linux board's SD card using an SD card reader connected to the development machine. In the following command substitute the appropriate string for 'SD_CARD_LABEL':

```
sudo cp arch/arm/boot/zImage /media/
SD_CARD_LABEL/
```

Compile the modules using the command

```
make modules
```

and then install them in a temporary directory:

```
make modules_install INSTALL_MOD_PATH=/
tmp
```

You can now copy the modules you have built from the temporary directory onto the SD card:

```
sudo cp -r /tmp/lib /media/SD_CARD_LABEL/
```

I²C tools

The I²C bus has been popular in the microcontroller world for decades. Compatible devices available range from analog-to-digital converters and I/O expanders to temperature sensors. To drive the I²C bus from Linux we first have to activate the right driver in the kernel configuration:

```
Device Drivers → I2C support → I2C
Hardware Bus support → I2C bus support
for Philips PNX targets
```

Measuring voltages more accurately

In the fifth installment of this series [15] we showed how to measure external voltages using the analog-to-digital converter in the microcontroller. Naturally many readers tried this out, and the unanimous response was that the 3.3 V supply voltage on the board, which is used as a reference by the converter, is not stable enough to obtain accurate readings. There is a simple solution to this, generating a more stable voltage by adding a 10 µF and a 100 nF capacitor in parallel to ground at one end of R19: see **Figure 18** and **Figure 19**. This should give a noticeable improvement in the accuracy of the conversions.
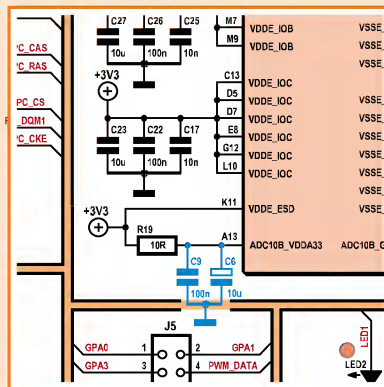
The driver is already included in the standard configuration.
Tools are available for Linux to carry out simple I²C bus tests. These have to be downloaded from the internet, unpacked and then written onto the SD card.
First download the I²C tools to the development PC

```
wget http://ftp.de.debian.org/debian/
pool/main/i/i2c-tools/i2c-tools_3.0.2-5_
armel.deb
```
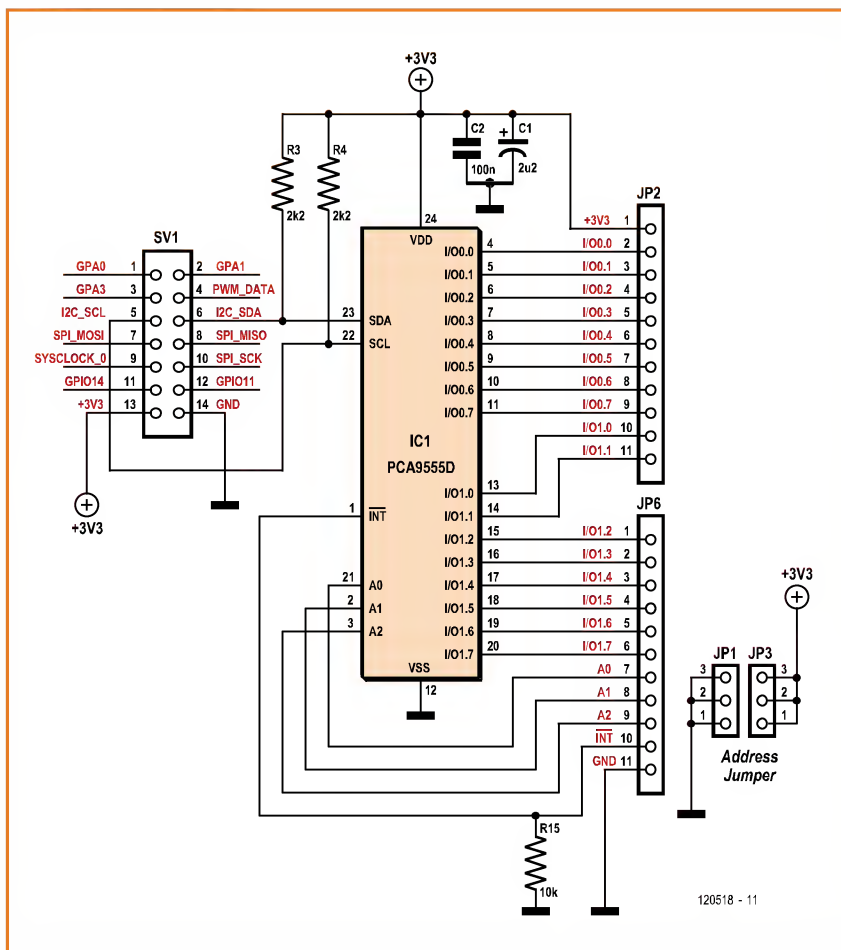
Figure 2.
Circuit for the PCA9555 port expander.

and then convert it to a standard archive:

```
alien -t i2c-tools_3.0.2-5_armel.deb
```

This will create a file in your current directory with the name 'i2c-tools-3.0.2.tgz'. Using the SD card reader, copy this file onto the card:

```
cp i2c-tools-3.0.2.tgz /media/
SD_CARD_LABEL/
```

After rebooting the Linux board the converted package archive has to be unpacked, and the I^{2}C device nodes need to be created. These actions only have to be done once.

First, then, we unpack the Linux I^{2}C tools:

```
cd /
```

```
tar xvzf i2c-tools-3.0.2.tgz
```

As before [6] we create the nodes in the file system corresponding to the devices using the 'mknod' command. To be on the safe side, find out the correct major number using:

```
cat /proc/devices
```

and check that the relevant entry reads '89'. If the value is different, the following commands will need to be amended suitably:

```
mknod /dev/i2c-0 c 89 0
mknod /dev/i2c-1 c 89 1
```

With the device nodes created and the software installed, we can connect our first I^{2}C device and try to talk to it. Simple tests can be carried out using a PCA9555 port expander chip, connected as shown in **Figure 2** to the 14-pin expansion header on the Linux board.
For the tests the device will need a 3.3 V power supply, which can be obtained from the expansion header. Jumpers can be fitted to inputs A0 to A2 of the port expander, allowing each to be taken to 3.3 V or to GND. The levels on these pins determine the address of the device.
Having connected the chip to the SDA and SCL signals on the *Elektor* Linux board, you can use the tool 'i2cdetect' to scan all the addresses on the bus:

```
i2cdetect 1
```

The result is shown in **Figure 3**. The address 0x6e is assigned to the LPC3131's interface itself for when it is used as an I^{2}C slave. Since all the address selection pins of the PCA9555 are taken high in our circuit, it responds to address 0x27. We can continue to test the function of the I^{2}C bus by wiring three LEDs to port 0 of the chip (see **Figure 4**). The command 'i2cset' can be used to send short I^{2}C bus messages using the command line. For example, to define all the port expander's I/O pins as outputs, type

```
i2cset 1 0x27 0x06 0x00
```

and then to set the outputs high type

```
i2cset 1 0x27 0x02 0xff
```

and the result should be that the LEDs light. Now that we have checked from the command

line that we can issue simple I²C commands, we would like to move on to controlling the bus from a program written in C.

I²C in C
The LPC3131 has two physically separate I²C interfaces, of which only the second is made available on the 14-pin expansion header on the board.

The device files created above corresponding to these interfaces are '/dev/i2c-0' and '/dev/i2c-1'. Accessing the hardware is done in the usual way: open the device file and write to it or read from it as needed.

Listing 1 shows a short C program that will light our LEDs. The program can be compiled on the development PC with the command

```
arm-linux-gnueabi-gcc -o i2ctest
i2ctest.c
```

and then the compiled code can be copied to the Linux board, either via the SD card or using the 'sshfs' or 'scp' tools. Alternatively, the program can be compiled directly on the Linux board using its built-in C compiler 'gcc'.

It is best to copy the compiled program into the '/usr/bin' directory so that it can be run from anywhere on the system without having to specify a path explicitly:

```
cp i2ctest /usr/bin
```

The program can then be run on the board, giving the device file for the I²C bus as its parameter:

```
i2ctest /dev/i2c-1
```

The LEDs on the expansion circuit should blink on and off.

In the first part of the source code in Listing 1 you can see how the open() function is used to obtain a handle for the device file. The ioctl() function, which is used to send configuration information to the device driver, is then used to set the address of the desired target device. Once the address is set, the write() function is then used to send a simple message over the I²C bus to the device. The byte sequence to set the port expander's pins to outputs is the same as we used above:
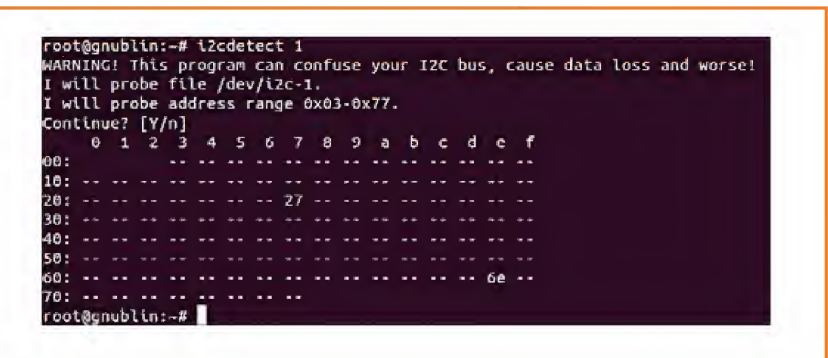
Figure 3.
Output of the 'i2cdetect' command.

```
buffer[0] = 0x06;
buffer[1] = 0x00;
```

The commands to turn the LEDs on and off are enclosed in an infinite loop, with a 100 ms pause between each call.

Other devices on the I²C bus can be communicated with in the same way. C experts may want to take a look at the 'libi2c' library [7], which provides a very flexible way of accessing the bus. But under the hood this library also just uses the same ioctl(), read() and write() functions.

Controlling serial ports from the command line
In the fourth part of this series [6] we installed a USB-to-RS232 converter. There are various ways in which serial interfaces can be integrated into a Linux system. For example, the microcontroller's built-in serial ports are usually accessed as '/dev/ttyS0', '/dev/ttyS1' and so on, with the default console, where system messages and the initial login prompt appear, usually being '/
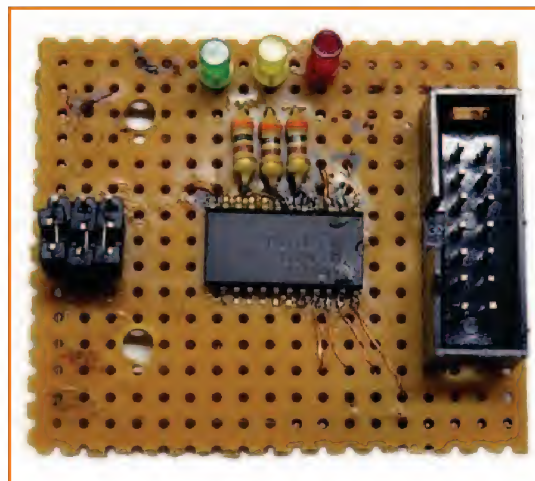
Figure 4.
Testing the circuit with three LEDs.

●Projects

Listing 1: Driving an I²C port expander

```
#include <stdio.h>
#include <fcntl.h>
#include <linux/i2c.h>
#include <linux/i2c-dev.h>

//Slave Address
#define ADDR 0x27

int main (int argc, char **argv)
{
  int fd;
  char filename[32];
  char buffer[128];
  int n, err;

  if (argc == 0) {
      printf(„usage: %s <device>\n", argv[0]);
      exit(1);
  }
  sprintf(filename, argv[1]);
  printf(„device = %s\n", filename);

  int slave_address = ADDR;

  if ((fd = open(filename, O_RDWR)) < 0) {
    printf(„i2c open error");
    return -1;
  }
  printf(„i2c device = %d\n", fd);
```

```
  //prepare communication
  if (ioctl(fd, I2C_SLAVE, slave_address) < 0) {
    printf(„ioctl I2C_SLAVE error");
    return -1;
  }

  write(fd, buffer, 2);

  //Port-0 GPIO as Output
  buffer[0] = 0x06;
  buffer[1] = 0x00;
  write(fd, buffer, 2);

  n = 0;
  while (1)
  {
    buffer[0] = 0x02;  /* command byte: write
output regs */
    buffer[1] = 0x00;  /* port1 data  */
    write(fd, buffer, 2);

    usleep(100000);
    buffer[0] = 0x02;  /* command byte: write
output regs */
    buffer[1] = 0xff;  /* port1 data  */
    write(fd, buffer, 2);

    printf(„%d\n", n++);
    usleep(100000);
  }
}
```

dev/ttyS0'. USB-to-serial interfaces appear as '/dev/ttyUSB0', '/dev/ttyUSB1' and so on. Nevertheless, whatever type of serial interface we use, the interface through which it is accessed is always the same.

The way Linux treats devices as files gives us a certain amount of platform independence: for example, we can develop and simulate an application that uses a serial port on the development PC, addressing the USB-to-serial converter as '/dev/ttyUSB0' just as we would on the Linux board. Then it is an easy job to transfer the software from the PC to the Linux board by cross-compiling it and then running it supplied with the correct parameters.

If a device file is missing it can be created using 'mknod' as before, assuming that the correspond-

ing driver has already been loaded:

```
mknod /dev/usb/ttyUSB0 c 188 0
mknod /dev/usb/ttyUSB1 c 188 1
mknod /dev/usb/ttyUSB2 c 188 2
```

In the fourth part of this series we used 'microcom', a simple terminal emulator program for communicating over serial ports. The desired baud rate and device are given as parameters to the command. When the program has been started any received characters are displayed immediately on the screen, and any characters entered on the keyboard are transmitted on the serial port.

However, we would like to be able to drive the port without using a ready-made program. Send-

ing and receiving data is not complicated: we can simply use normal commands to write to and read from the device file '/dev/ttyUSB0'. However, setting the baud rate of the port requires using a special technique.

Under Linux the command we need is called 'stty'. For example, to set the port speed to 38400 baud, the following command must be issued:

```
stty -F /dev/ttyUSB0 38400
```

We can now send a simple string of characters like this:

```
echo "Hello world" > /dev/ttyUSB0
```

for this program. The first part of the program sets up the basic configuration using the data structure struct termios options. The function tcgetattr() first writes the current configuration into the data structure so that it can subsequently be modified. The data rate is set using the functions cfsetispeed() (for the input data rate) and cfsetospeed() (for the output data rate). At this point it is also possible to modify the number of data bits, start bits and stop bits. Finally the new configuration is put into action by passing it to the driver using the function tcsetattr().

Sending and receiving data works in the same way as with any other standard device. Using the

and receive characters like this:

```
cat < /dev/ttyUSB0
```

You may want to experiment a little with these commands, which are more than adequate for the manual testing of communications with simple protocols that only involve printable ASCII characters. However, if non-printable characters are involved things get more complicated, and a short C program will be an easier way to go.

Controlling serial ports from a C program

It is a relatively small step to move to using a C program to control a serial port. As we saw above when driving the port from the command line we must first configure the interface, the most important configuration parameter being the baud rate. In the download accompanying this article [8] is a C program (called 'Listing 2') that shows how things are done. The examples from [9] were used as the model

function write() (in conjunction with a handle previously obtained from a call to open()) you can send data, and using read() you can receive data. In the example here the connected device is expecting a command six bytes in length: as you can see the variable buffer[0] is set to zero: this value is not so convenient to generate from the keyboard.

The program expects a reply from the connected device, also six bytes in length. Reception is done in a loop which assembles the six characters as they are read in. The program waits at the function (a 'blocking' call) for each character to arrive. This is not the ideal solution, as it limits the data rate to 19200 baud. The microcontroller may be fast and have little to do in the loop except wait idly for the next character, but nevertheless we soon run out of precious processor time.

The usual solution to this problem is to use an interrupt. When a data byte is available the processor is notified and the character is fetched from the receive buffer. Under an operating sys-

Author Benedikt Sauter in an Elektor interview. There were lots of cameras in action on the Farnell/ element14 stand at electronica 2012. You can watch the video at [13].
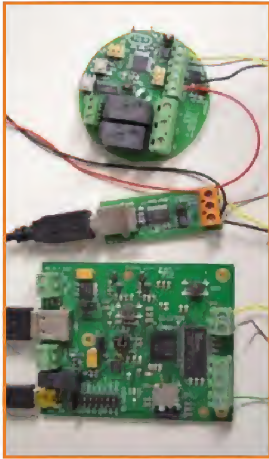
Figure 5.
RS485 connection to the
ElektorBus.
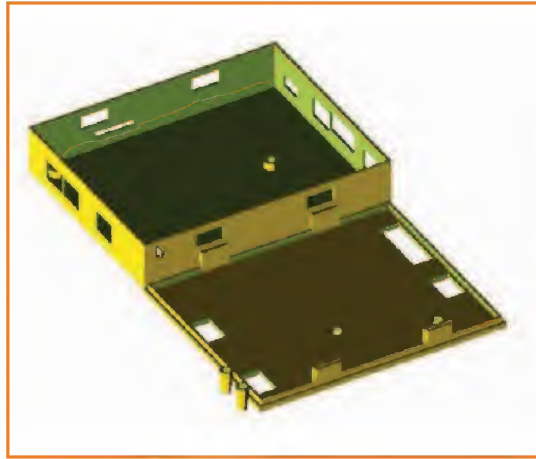
Figure 6.
Elektor Linux board enclosure.
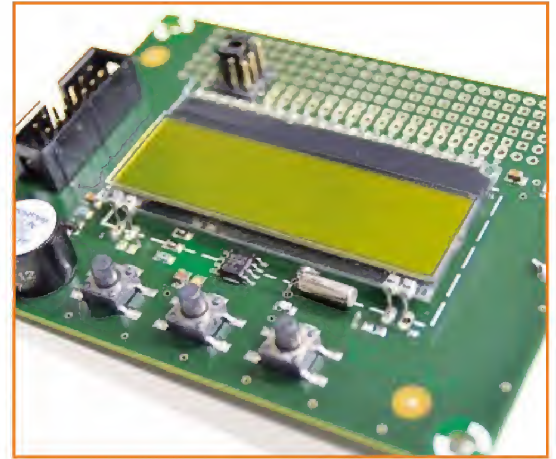
Figure 7.
The expansion board that we will be describing in the
next edition.

tem like Linux this can result in a program that makes efficient use of resources. If no data bytes are available, the receiving program is free to get on with something else or allow itself to be descheduled; when characters arrive the program is woken up to process the data.

The third program in the download directory [8] uses interrupts in this way. It is essentially the same as the example above, but includes a new function signal_handler_IO() which is called only when data bytes have been received.

RS485 and the ElektorBus

Now that we have worked out how to drive a serial port, we should also be able to send and receive data over an RS485 interface. It should just be a matter of replacing the USB-to-RS232 converter on our board's USB port with a USB-to-RS485 converter. A suitable converter can be ordered from *Elektor* (order code 110258-91) [8]. This device is based around an FTDI chip, which already has a suitable driver in the kernel that simply needs to be activated, as described in the fourth installment of this series.

The remote device on the RS485 bus for this experiment is an ElektorBus relay module [8] [10], which includes two relays (**Figure 5**). Listing 2 needs to be modified slightly to implement the ElektorBus protocol, for example to change the transmission speed to 9600 baud and the variable LENGTH to 16. Before sending a message the array Buffer must be populated with

the sixteen bytes that are to be sent. To control relay 1 on the ElektorBus board, send the following commands:

```
170,0, 0,5,0,10, 96,1,0,0, 0,0,0,0, 0,0
(relay on)
170,0, 0,5,0,10, 96,0,0,0, 0,0,0,0, 0,0
(relay off)
```

For relay 2 the sequences are:

```
170,0, 0,5,0,10, 0,0,96,1, 0,0,0,0, 0,0
(relay on)
170,0, 0,5,0,10, 0,0,96,0, 0,0,0,0, 0,0
(relay off)
```

More about the ElektorBus can be found at [11].

What the future holds

If you are looking for a suitable enclosure for the *Elektor* Linux board (**Figure 6**), the internet is your friend [12]. At the link you will find a design that can be printed using a 3D printer. The enclosure is designed not to require any screws for assembly: the lid and the base simply snap together.

This is the last installment in our embedded Linux course, but it is certainly not the end of the project. A new board (**Figure 7**) is in the works to extend the *Elektor* Linux board with many handy functions:

- display (two rows of sixteen characters);
- three buttons to provide a user interface or menu control;
- sixteen extra digital inputs and outputs;
- a real-time clock with battery supply;
- a buzzer to provide acoustic indications;
- a prototyping area for additional circuitry.

In the next edition we will describe this board in detail.

(120518)

Internet links

[1] sauter@embedded-projects.net

[2] www.gnublin.org

[3] www.elektor-magazine.com/120180

[4] www.elektor-magazine.com/120578

[5] http://en.gnublin.org/index.php/Eclipse

[6] www.elektor-magazine.com/120181

[7] http://opensource.katalix.com/libi2c/

[8] www.elektor-magazine.com/120518

[9] www.tldp.org/HOWTO/
    Serial-Programming-HOWTO/

[10] www.elektor-magazine.com/110727

[11] www.elektor-magazine.com/elektorbus

[12] www.thingiverse.com/thing:29314

[13] www.element14.com/community/
    community/events/electronica

[14] http://eagleup.wordpress.com/

[15] www.elektor-magazine.com/120182

Arduino on Course (4)
A plant watering supervisor for communal use

By **David Cuartielles** (Sweden/Spain)

At Elektor's recent Expert Meeting a full day was spent discussing how we make projects, or which are the hot topics we think Elektor should cover in the future. During one of the round-table sessions, E-Editor Jan suddenly started dropping small piles of components on our tables and asking the question: *"Picture this! A weekend off, you start rummaging your e-junkbox and hit upon these components. What would you do with them?"*

When I returned home from that trip I started to think about those forgotten components I own. I have plenty of things bought during the experimental phases of projects that never made it into the final prototype. I opened one of the boxes and I found a dot matrix display and a couple of buttons. I thought it would be interesting to make some sort of clock out of it. So I bought a Real Time Clock (RTC) chip from a local supplier and hooked everything together. The result is pictured above, and described below.

Materials
The materials for this month's hands-on experiment are:

- Arduino Uno board
- Prototyping shield
- Pinheaders (male)
- USB cable
- Two pushbuttons
- Dot Matrix display w. HT1632 controller; I used the 32x16 from Sure
- RTC DS1302
- 32 kHz quartz crystal
- CR2032 battery holder
- 9 V battery connector
- 9 V battery (for Arduino) + CR2032 battery (for the RTC)
- A box to put everything in

Note: I built this project in a cardboard box, but I reckon there should be many other creative housings out there ranging from a discarded lunch box to a more professional plastic enclosure… everything goes!

My design brief: make something useful
I gave some thought to what I could do with my salvaged parts. I live in an apartment building and have some shared plants in the staircase well. Meaning: one of the apartment dwellers waters those plants rather unsystematically, so nobody knows for sure when the plants are okay for water. The truth is that most of the plants that appear in the staircase space end up drying up sooner or later.
I'd like to believe people do not water those plants because they think others are doing it, and not because they don't care. So I came up with the idea of making a time counter that supplies a

visual indication of the last time the plants got watered by "somebody".

I didn't want the project to be too complex, therefore I decided that it should be focusing only on the human aspects of watering a plant and it should not involve measuring the humidity of the plant's soil or any other ambient conditions around the plant itself. Neither did I want to make anything connecting to the Internet to remind me about watering the plants (the project Botanicalls [1] is doing precisely that, it tweets a message when the plant is in dire need of water).

I just wanted a device to let my neighbors check out when was the last time someone took care of the plants. So I thought about the chess clock paradigm. Players have to press a button on the clock at the end of each move to pass the turn to their opponent. My idea was to create a clock that would offer the same interaction. Once someone waters the plant, he or she presses a button and passes the responsibility to someone else. When pressing yet another button, the machine will indicate just how long the plant hasn't been watered.

At this point, I mustered the components I figured I'd need (**Figure 1**).

Controlling the dot matrix display

There is something about blinking LEDs that fascinates people. It is very easy to make the connection between controlling a single 5 mm LED like the one we all use in small projects and a huge display in a public space. I bet that, as soon as you got your first LED to shine, the possibility of putting a hundred together to do the same starts milling around in your head.

There is a limit to the number of pins on a microcontroller, and another limit on how much current each pin can supply. A Dot Matrix display is a plastic unit in which the LEDs have either their anode or cathode commoned, which makes them easier to handle mechanically. Sometimes driver chips are included as part of the definition of Dot Matrix display. For our project the various parts are connected up like in the "schematic" in **Figure 2**.

I own a couple of displays from a brand called Sure. I got them from the Arduino Store for some experiments I did for my classes last year. I am sure you can get the same display from other vendors as well. It was documented a-okay online, so I thought it would be good to add it to my repertoire.

This display controls 16x32 R/G LEDs. In other words, it has 512 bicolor LEDs. They can be red, green or both at once, which gives orange light. It is also easy to daisy chain and I have read about projects that control up to four of these displays in a row, that is 2048 LEDs!

For this project I use only one of the LEDs as I don't need to show that much information and I just want to display how much time passed. But you may want to do more, so let's take a look at what you can do with the display.

Figure 1.
The parts that go into the Arduino'd Plant Watering Supervisor.
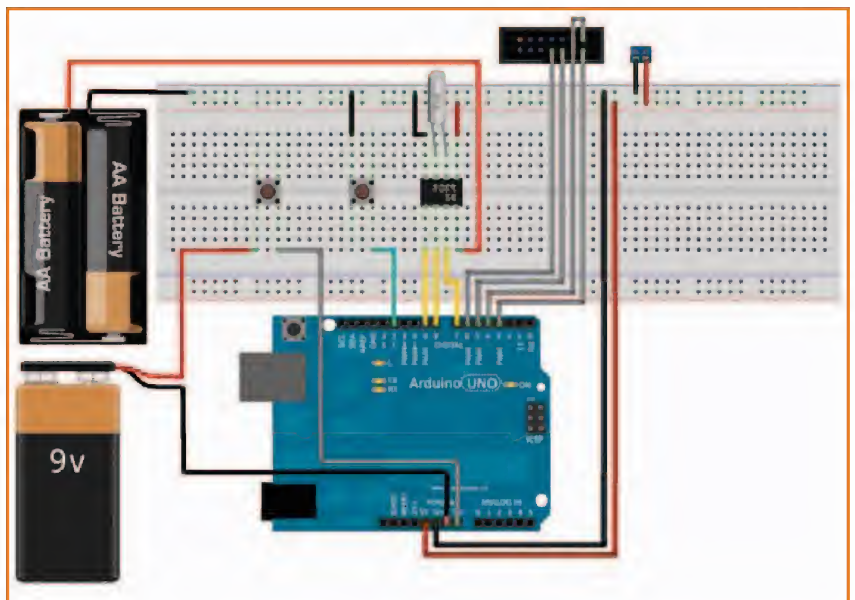
Figure 2.
Schematic showing the whole project.

Install the library for the display

I put together a library using code from several authors, just re-factored the code to respect the Arduino library style, and added a couple of simple examples. The library is compatible with revisions of the IDE 1.0 and above. Everything is aimed at using an RTC (real time clock) unit built on the Arduino prototyping shield, see **Figure 3**.

You will have to download the library from Elektor's website [4] and install it. Remember that adding a new library to the IDE is effectively done by creating a folder called 'libraries' inside your sketchbook and uncompressing the file you got from the website right there. After that, you should restart the Arduino IDE and the library will then show up in the above mentioned menu. Note: Arduino's IDE is going to include a system

Listing 1.

```
#include <fonts.h>
#include <HT1632c.h>
#include <images.h>

HT1632c display(6, 5, 3, 4);

void setup() {
  display.setup();
}

void loop() {
  display.text("Hej", 5, 5);
}
```

Listing 2.

```
#include <fonts.h>
#include <HT1632c.h>
#include <images.h>

HT1632c display(6, 5, 3, 4);

void setup() {
  display.setup();
}

void loop() {
  display.image(Arduino_logo, 0, 0, 32, 16);
}
```

Listing 3.

```
#include <fonts.h>
#include <HT1632c.h>
#include <images.h>

#define heart_icon_width 17
#define heart_icon_height 16

unsigned char PROGMEM heart_icon[] = {
   0x00, 0x1e, 0x00, 0x3f, 0x80, 0x7f, 0xc0, 0x7f, 0xe0, 0x7f, 0xf0, 0x7f,
   0xf8, 0x3f, 0xfc, 0x1f, 0xfe, 0x0f, 0xfc, 0x1f, 0xf8, 0x3f, 0xf0, 0x7f,
   0xe0, 0x7f, 0xc0, 0x7f, 0x80, 0x7f, 0x00, 0x3f, 0x00, 0x1e };

HT1632c display(3, 4, 5, 6);

void setup() {
  display.setup();
}

void loop() {
  display.image(heart_icon, 0, 0, heart_icon_width, heart_icon_height);
}
```

to automatically install libraries from the compressed files you download from the Internet. However at the time of writing this feature isn't deployed yet.

With the library you will be installing a series of examples that are educational to follow in detail, even if you do not envisage to replicate the hardware in detail. The examples will allow you to:

- test whether the display is working properly;
- show simple text messages;
- scroll text on the screen;
- load images.

It's easy to access the examples, just use the menu to navigate through: *File / Examples / HT1632c*.
Whenever you want to use the library, you will need to include three header files: fonts.h, HT1632c.h and images.h. When calling the constructor you will need to specify the pins you have attached the display to. They have the following order: Data pin, Write clock pin, Chip Select, and Clock pin.

Configure the display
In this project, the display is connected to digital pins 3, 4, 5, and 6. The library allows any digital pins to be configured to send data to the LEDs. You can also daisy chain multiple displays in sequence. I only had two to test, but I am confident it will work for more, without problems. The display is powered directly from the Arduino regulator; during my tests I didn't seem to need significantly more power than the battery was able to source. If you are planning on connecting many displays as part of one single project though you should keep in mind the amount of current you will need for the system to keep working.
Now, **Listing 1** configures the display with the

'simple_text' example that comes with the library. By default the text method will be using orange as a color to show the text. A fourth parameter will determine the color, you can use the constants: BLACK, GREEN, RED, and ORANGE.

Show a simple image
It is possible to render low resolution images on one of these displays. You will need to have the image stored in program memory as an array
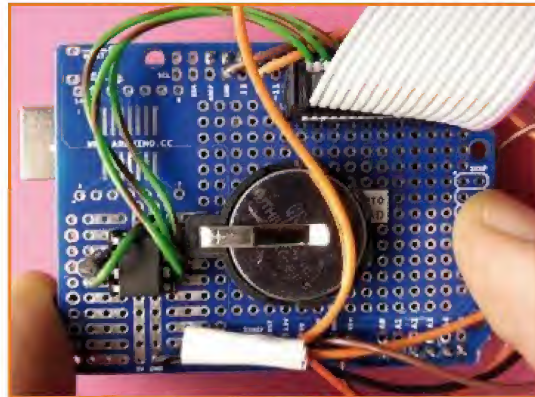
Figure 3.
The Arduino prototyping shield with the components soldered.

Figure 4.
Wow! Our 'screen' is showing the Arduino logo from the 'Simple_image' example.

like in **Listing 2**, employing the 'Simple_image' example, loading a default icon.
Figure 4 shows the result of loading the Simple_image example on the dot matrix display or is it "the screen"? In this case it is the Arduino logo type I stored in the library as a default example. If you want to display your own image, you can generate code friendly images using Gimp, which is a free image editing suite that runs on all the main operating systems. You will need to:

- open the image and scale it to fit the screen (32 columns x 16 rows);
- rotate the image 90 deg. clockwise;
- export them as XbitMap;
- open the xbm file and copy the resulting array;
- paste the array in your program as unsigned char PROGMEM, in other words, push it into program memory space, just to leave the RAM untouched.

An example of a result is shown in **Figure 5**, it's a clipart of a heart, and importing it into code would look like **Listing 3**.

Figure 5.
Clipart of a heart image
before being processed for
inclusion in the Arduino
Sketch]

Note: the way the xbm file is rendered is a little different from the way it gets rendered on the screen. Therefore you need to change width for height when you want to load your own icon. In my case, the icon was 16x17, but I enter the data as 17x16 (width x height).

Tip: you can add multiple icons/images to your program, just give them different names so that you can call them from anywhere in your code. There is a lot that can be done with the library controlling the display, but for the purpose of making this project, we just need to show text and possibly some simple images. Feel free to explore the library as it is free software, and the functions in it are self-explanatory. Let's move on and explore how to use the Real Time Clock.

Real Time Clock chips

Thanks to an external 32 kHz crystal RTCs can measure time in a very precise way. As RTCs need to be counting time constantly, they can-

not be switched off, and therefore we need to power them with a battery. On the other hand, they consume very little power and can run on 3.3 V batteries for a really long time, sometimes even years.

The chip I picked for my project is the DS1302 from Maxim, a very common RTC that's been widely documented by the Arduino community. There is a really good read available by Arduino user Krodal at the Arduino Playground [2] together with some code exploring the capabilities the chip in depth. However, for the sake of simplicity, I decided to use a library by Matt Sparks [3].

So far we have only used four pins on the Arduino Uno to connect the display; the RTC needs three digital pins, so I chose 7, 8, and 9. **Figure 6** shows the RTC, where to plug in the power and the 32 kHz clock, as well as the pins where to connect the chip to Arduino.

Install the RTC library, set the time

You can get a copy of the RTC library from the Elektor website [4] or from the github project by its author [3]. Install it by uncompressing the file inside your libraries folder, as explained earlier for the HT1632c library, and check inside the examples.

You need to run the example set_clock.pde once in order to configure the time on your RTC. As long as the RTC itself is plugged to a battery, you won't need to run this code again. Make sure you modify the line inside setup that configures the time in the clock, **Listing 4** shows the part of set_clock.pde you need to configure manually.

Use the EEPROM to store current time

The ATMega 328 on the Arduino Uno has 512 bytes of EEPROM, or non-volatile memory, to retain data when the system is off. This part of the memory is frequently used by embedded designer to store the basic configuration of a device.

In our case, we will use the EEPROM to store the date when someone last watered the plants (and pressed the button). In total we need to store three bytes: year, month, and day. Then we need to add to our program the ability to estimate the time difference between the time stored in memory and the one we could read at any time, as we want to show on the screen that the plant hasn't had any water for some days. **Listing 5** shows how to store something in EEPROM.

Figure 6.
Schematic showing the RTC connected to Arduino.
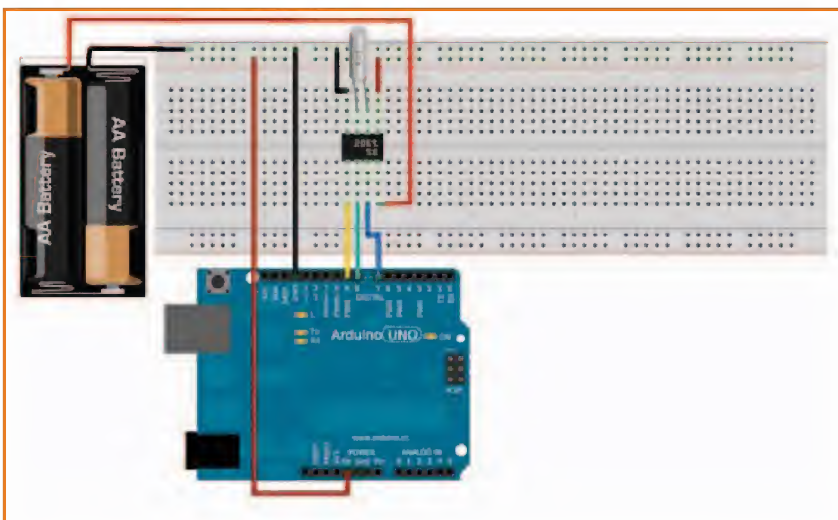
```
Listing 4.

[…]

void setup() {
  Serial.begin(9600);

  /* Initialize a new chip by turning off write protection and clearing the
     clock halt flag. These methods needn't always be called. See the DS1302
     datasheet for details. */
  rtc.write_protect(false);
  rtc.halt(false);

  /* Make a new time object to set the date and time */
  /*   Tuesday, May 19, 2009 at 21:16:37.             */
  Time t(2009, 5, 19, 21, 16, 37, 3);

  /* Set the time and date on the chip */
  rtc.time(t);
}

[…]
```

A project made from things
I found in my e-junkbox

A button to store time

A button supplies a trigger signal flagging that a time needs to be recorded. When the button is pressed, the current time has to be read from the RTC, and pushed to the EEPROM. I associated the button with pin 12, in my code (**Listing 6**) I will make sure that pin is configured as INPUT_PULLUP just to save some time when soldering the components together.

After storing the time, the readout on the box will show the message "nice!", and stop there. Other operations are blocked until the program is restarted. I think this is a good workaround to stop people from reading and writing to the EEPROM too often.

Check the full code listing in the Elektor archive file for this article. I have added an extra animation to thank my neighbors for watering the plants!

A button to control it all

Probably the most important part of this proj-

ect is the way battery consumption is handled. I decided that the best way to save battery power is not to have the system on at all. As you might have noticed in the bill of materials for this project, there are two pushbuttons listed in it. And so far in the code we are only using one.

```
Listing 5.

#include <EEPROM.h>

int val;

void setup() {
}

void loop() {
  val = analogRead(A0);
  EEPROM.write(0, val);
}
```

Figure 7.
"Hey, it's been three days since I last got some water".

The second button is all about cutting the battery power; unless you press it, the system is not going to get any power. Microcontrollers wake up rapidly; in this case, the first thing you will see on the screen is the Arduino logotype and, after a few seconds, it will tell you the number of days since the plant was watered.

Adding this button also means that, for the system to store the new time, you need to press both buttons at once. One will turn the system on, while the other will tell the system to store the time from the RTC in the EEPROM. Like in **Figure 7**.

Closing words

With the excuse of building 'something' from 'any' parts, we got the chance to experiment with drivers capable of driving hundreds of LEDs, as well as with timers. The next step is going to be checking out whether my prototype fits the purpose. I will place it in my staircase and see if 'our' plants get a better life.

I am sure you can figure out multiple uses for a box with a display and two buttons beyond the one described in this article: from clocks to games. Just add a buzzer and some more pushbuttons to the mix and turn it into a small game console.

(120714)

References

[1] The Botanicalls project:
    http://botanicalls.com

[2] Krodal's explanation on the DS1302 RTC:
    http://arduino.cc/playground/Main/DS1302

[3] Mark Sparks' RTC library:
    https://github.com/msparks/arduino-ds1302

[4] Program and library downloads:
    www.elektor-magazine.com/120174

Acknowledgements

Listing 6.

```
[…]

int saveTimePin = 12;
int saveTimeButton = LOW, saveTimeButtonOld = LOW;

void setup() {
  […]
  pinMode(saveTimePin, INPUT_PULLUP);
}

void loop() {
  […]
  saveTimeButton = digitalRead(saveTimePin);
  if(saveTimeButton == LOW && saveTimeButtonOld == HIGH) {
    // we will use the Time stored in t to save the time
    EEPROM.write(0, t.yr);   // position 0 — year
    EEPROM.write(1, t.mon);  // position 1 — month
    EEPROM.write(2, t.date); // position 2 — day
    // clear the display
    display.cls();
    // show a thank you text
    display.text("nice!", 2, 5);
    // stay here
    while(true) {};
  }
  saveTimeButtonOld = saveTimeButton;
}
```

The **7-uP** Alarm Clock/ Time-Switch

Part 1

By
Michael J. Bauer
(Australia)

Although the main objective of the 7-uP project was to create a digital alarm-clock with enhanced functionality, a key feature is the ability to automate the operation of appliances in the owner's study, office or bedroom, controlling devices such as lamps, computer & peripherals, and audio-visual equipment. Some enthusiasts might also think of some ingenious ways to apply the device to wake themselves up on work-day mornings!

This project all began out of frustration with my alarm-clock/radio, which despite being a big-name brand, and not cheap, is very awkward to use and is lacking most of the features I would prefer to have. Whenever I raise the issue among friends and work colleagues, it seems that all alarm-clocks (and clock-radios) fall short of user expectations in some way or other. Here's a summary of complaints I've often heard from people about their alarm clocks...

- Forgetting to switch on the alarm before going to bed (because it had to be switched off in the morning to silence it, of course).
- Clock-radio not tuned into a station properly, or the volume was turned way down low, so the alarm could not be heard when it activated.
- Tedious ritual to change the clock time or alarm time, typically using one button to set 'hours' and another for 'minutes'. Holding down a button (in some cases two) causes the minutes or hours displayed to change automatically, but in most cases either too slow or too fast.
- Alarm time is the same every day, whether it's a weekday, Saturday or Sunday. (Most people need to wake up earlier on weekdays than on weekends and some people work part-time or shifts varying each day.)
- The snooze time interval is fixed (typically 7 to 9 minutes) and can't be adjusted.
- It's a pain having to press the snooze button to silence the alarm. It would be great to have the option for the alarm to sound for a few seconds, then go silent for a few minutes, and repeat the process until canceled somehow.
- The display is too bright at night or too dim in daylight.
- There's a switch to adjust the display brightness, but it's awkward to use. Why can't the clock adjust its brightness automatically to suit the ambient light level?

DIY cannot compete with mass-production, but there is great satisfaction derived from making something better

- I don't like waking up to the radio in the morning, but isn't there an alarm-clock that makes a more interesting sound? (Or, better still, a choice of different sounds.)
- Toddlers like to play with the buttons on the alarm-clock. Often the time or alarm settings get changed accidentally.
- The snooze button is too small, or badly located, or otherwise difficult to operate in the dark.
- It doesn't make the coffee.

How many of the foregoing complaints apply to your alarm-clock?

A trip to the local electronics store failed to find an alarm-clock meeting enough of the requirements on my wish list. So, being an Electronics Engineer by profession, I decided to design and build my 'ideal alarm-clock'.

An improved digital alarm-clock design

Following is a concept for a digital alarm-clock meeting my objectives...

My main focus was to combine enhanced functionality with convenience (ease of use), while attempting to address many of the deficiencies of commercial products. The design would not be compromised to compete on a cost basis with consumer mass-market products. Do-it-yourself construction cannot compete with mass-production, but there is great satisfaction derived from making something better.

An intuitive local user interface (control panel) comprising display, a number of push-buttons and a rotary encoder switch (knob) should be provided for operation of the clock while not connected to a computer. The local controls would be designed primarily for alarm time setting and for selection of displayed information, rather than for advanced configuration. Programming of the 7-day time-switch ON/OFF schedule, if required, would be more conveniently done via a USB PC link using a Windows software application.

The front panel should have a large 4-digit display, normally showing the time-of-day, plus several dedicated back-lit enunciators, e.g. PM,

24 hr, A1–A4, S1–S4, MON, TUE, WED, etc. The 'alarm' enunciators (A1, A2, A3, A4) show the ON/OFF status of up to four daily alarm events. Another set of four enunciators (S1, S2, S3, S4) indicates the ON/OFF status of the time-switch outputs.

Display brightness must be variable, and preferably controllable by a choice of user-configurable options, e.g. ambient light sensor ('automatic'), or alarm event ('power-save mode'), or manually using the control panel.

The alarm function should have a variety of configurable options, e.g. choice of alarm sounds, initial volume, maximum volume, volume auto-increasing ('ramp-up' option), number of repeat alerts, interval between repeat alerts, etc, etc.

There should be a good selection of preset (built-in) alarm sounds, which can be customized by the user. A different alarm sound could be programmed for each alarm 'event' occurring throughout the 7-day alarm schedule, if the user so desired. Ideally, alarm sounds would be digital audio clips (PCM/MP3 encoded), downloadable via the USB link, giving wide-range low-distortion audio quality. (This could be an optional extra, as the added cost might not be justifiable to many enthusiasts.)

The 'snooze' function might have two modes of operation. In 'classic' snooze mode, the alarm sound is silenced temporarily by a single push of the button, but is reactivated after a preset time interval (e.g. 10 minutes). Alarm activation repeats until canceled (e.g. by pressing the snooze button three times rapidly) or until a preset timeout expires.

In the alternative 'auto-repeat' snooze mode, however, alarm activation would recur automatically at preset intervals (e.g. 10 minutes), for a number of times, without operator intervention, before canceling itself. A single press of the snooze button would cancel repeat activations, regardless.

In either snooze mode, it would not be necessary to switch off the alarm every morning and re-enable it again before going to bed every night!

In normal operating mode, the knob would be a manual brightness control for the display. In

time-setting and configuration modes, however, the knob could well be used for convenient adjustment of displayed data.

A means to confirm the next pending alarm event, up to 48 hours in advance, must be provided. Perhaps pressing the snooze/cancel button (while there is no alarm active) could enter an 'alarm confirmation' mode in which the time and day of the next pending alarm is displayed. And if the button was held down for longer than (say) two seconds, the sound assigned to the pending alarm event would be played to check the loudness level. Taking this concept a step further, the knob could be used to adjust the loudness, while a pair of buttons might be used to change the sound effect.

For technology enthusiasts, the clock should connect to a PC (via USB) and support a variety of accessories via rear-panel I/O connectors. These provide logic outputs which can be used to switch appliances on and off at preset times, and/or to switch on when the alarm activates. Examples of accessories are: alarm bell/chime, bed-side lamp, radio, TV, kettle, coffee-maker, computer, PC peripherals, etc. Each of the time-switch 'channels' should be settable independently and may have ON/OFF times set differently for each day of the week.

A backup battery must be provided to maintain the clock time and date in the event of temporary power failure. When the battery voltage drops below the minimum usable voltage, a warning should be indicated.

All user-settable options, parameters, on/off/alarm weekly time schedule, etc., must be preserved in non-volatile memory. User configurations may be transferred to and from a PC using a Windows utility. While connected to a host PC, the clock should automatically synchronize its time and date to the PC's internal real-time clock, which in turn may be synchronized to Internet time.

In addition to supporting a Windows 'graphical user interface' (GUI) to configure clock options, and for programming the alarm schedule, etc., the USB port must allow downloading of firmware upgrades. This capability is essential to support feature enhancements and bug fixes.

Technical introduction

Essential hardware consists of a main board and display board, plus encoder switch (knob), speaker and snooze/cancel button. Surface-mount (SMT) components have been eliminated from the 2 boards comprising the basic hardware, to make assembly easier for hobbyists with lesser soldering skill and/or limited tool kit. The microcontroller chip (MCU) is socketed (52-pin PLCC) for the same reason.

The main board incorporates a simple, low-cost sound generator. Audio level (attack/decay amplitude envelope and overall volume) is controlled digitally by the MCU using pulse-width modulation (PWM). With the help of some tricky firmware, which can add frequency-modulation and/or amplitude-modulation effects, this arrangement can produce a good variety of alarm sounds ranging from quite pleasant to unbearably horrible (to suit all tastes). A selection of 16 pre-defined sounds is provided with the generic firmware. Alternative sounds may be customized by the user, by modifying the 'sound shape' parameters stored in EEPROM.

The parts may be housed in a ready-made enclosure, or in a unique housing of your own design and construction. The 'standard' enclosure is PAC-TEC model CM6-225, but a Chinese-made replica is available at a lower price. The circuit boards were designed to fit in this box. (*See Assembly Instructions in download package.*) Although the seven 'control surface' push-buttons are normally fitted to the top of the display board, so that they are accessible along the front edge of the top panel, they may be located elsewhere, e.g. on a piece of prototyping board, wired back to the display board. Such an arrangement would allow creative variations from the 'standard' enclosure design. Hardware expansion is possible via an optional 'mezzanine' board which connects to the main board via a ribbon-cable connector. This makes it possible to extend the functionality of the clock, or to use the hardware platform for a completely different purpose. A mezzanine board to generate hi-fidelity PCM/MP3 digital audio alarm sounds may be developed by the author in the future. The board will include an MP3 decoder chip and serial data-flash memory to hold alarm 'sound clips'.

The firmware is based on the author's 'ALERT' (Low-End Real-Time) operating system. Project builders may choose to write (and share) their own firmware, or to download a generic version free-of-charge. The source code is available at [1] for readers who may wish to modify or extend the firmware for their own purposes.

The hardware is based on Atmel's AT89C5131 USB microcontroller. This MCU was chosen

Features of the '7-uP' Alarm-clock/Time-switch

- Convenience and ease of use
- Intuitive local controls and setup menu
- Knob (rotary encoder switch) for easy time setting and data entry
- One-touch confirmation of pending alarm time and loudness
- Multiple alarm 'events' and weekly (7-day) schedule
- Program up to four alarm 'events' in a 24-hour day
- Different alarm times each day of the week, if desired
- Versatile snooze/cancel options
- 'Classic mode' —
  alarm sounds continuously until deferred by button hit
- 'Auto-repeat mode' —
  alarm recurs at periodic intervals, until canceled
- Adjustable snooze time interval (in either mode)
- Variable brightness display,
  with choice of control modes, i.e.
- Ambient light sensor
- Alarm or timer activated
- Manual override — adjust brightness using knob
- High quality, programmable alarm sounds
- Choose from a variety of 'interesting' preset alarm sounds
- Customize the preset sounds or create your own sounds
- Assign a different sound to each alarm 'event'

- USB link to host PC #
- Synchronize the clock to PC system time and date (i.e. Internet time)
- Save and restore option settings, alarm event and time-switch schedules ^
- Download firmware upgrades (feature enhancements and bug fixes)
- Time-switch and Alarm-switch accessory connections
- Four independent time-switch channels with 7-day ON/OFF schedule
- Control AC appliances (lamps, radio, TV/AV, computer & peripherals)*
- Alarm-activated output for controlling external wake-up devices
- Auxiliary (countdown) timer with accessory control output
- Switch PC peripherals on automatically when USB bus power is detected

* *Requires optional 4-outlet solid-state relay board for controlling AC line powered equipment.*

\# *PC link uses the USB-CDC Virtual COM port API, also accessible using HyperTerminal.*

^ *Facilitated by Windows GUI application software, to be developed by 3rd-party developers.*

---

because it is available in a PLCC package (socketable), it has a generous amount of on-chip flash program memory (32 KB) with separate data EEPROM (1 KB), it has an on-chip USB peripheral module, it is popular and hence readily available at low cost (around US$7.00 from Digi-Key). Most importantly, the 'C5131 has a bootloader for flash programming via the USB port, requiring no additional programming device.

The 'C5131's only bad trait is that it is has an 8051 core, which is (in the author's opinion) one of the worst MCU architectures ever devised! Not that it really matters if you're programming in C and there's enough flash PROM to compensate for the inefficient instruction set. (To be fair, the 8051 instruction set is quite efficient when working with the core 'register file', i.e. RAM below 0xFF, but horribly inefficient working with extended data RAM.)

For DIY firmware developers, there is a free C compiler available for 8051 code development. (Google with "SDCC 8051 compiler".) Overall, I think the 89C5131 was a good choice for a DIY project, especially for enthusiasts not skilled in

SMD soldering. For a commercial product of similar design, I would choose a different MCU, e.g. Freescale MC9S08-JM60, Microchip PIC18F66J50, or maybe even a 32-bit ARM core MCU (since the cost is comparable with many 8-bitters). All components are readily available from the major online suppliers such as Farnell / Avnet), Digi-Key, Mouser, etc. Most parts should also be stocked by local hobby electronics stores (probably at more inflated prices). Beware of high delivery fees when ordering online from overseas suppliers.

General operation
Power supply requirements
The clock is designed to be powered from a 9 V DC plug pack. The clock itself draws less than 100 mA (with the display set to normal brightness). A switch-mode regulated plug-pack is recommended to minimize AC power consumption. The clock has a switch-mode 5 V regulator IC for optimum efficiency.

The clock may also be powered from the USB port (V_{bus} = 5 Vdc). The external DC supply ($+V_{EX}$), normally available at the accessory sockets, comes directly from the 9 V power supply

●Projects
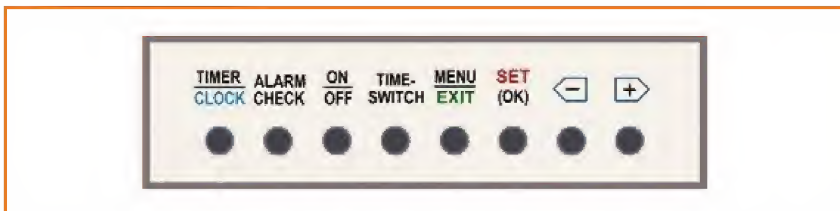
input, so it is not available when running on USB power. Consequently, accessories which rely on the 9 V DC supply will obviously not function. The backup battery keeps the clock running while disconnected from both USB and the 9 V plug-pack.

Backup battery

The clock is not intended to be battery powered in normal operation. The battery is intended only to prevent loss of time and date during the occasional brief AC line power failure; also to allow the clock to be disconnected momentarily while being moved from one location to another, e.g. for connection to a host PC. While running solely from the backup battery, only the bare minimum clock functions are kept alive to maintain the time and date.

The backup battery comprises three 'AA' size cells, which may be 1.5 V alkaline types or 1.2 V rechargeable types. The microcontroller runs on any battery voltage in the range 3.3 V to 5 V. Battery voltage is monitored continuously by the microcontroller when running on the main 9 V DC supply. A switch is provided to disconnect the backup battery in case the clock may be powered down for a prolonged time.

Accessory DC supply

The Alarm and Time-Switch Accessory sockets

provide a protected DC supply, fed from the 9 V DC inlet (plug-pack). The external DC sup-

ply is current limited to about 0.5 A by a resettable fuse. The alarm-switched control output is capable of sourcing up to 0.5 A to a DC-powered accessory (e.g. bell, lamp, radio, etc.).

Firmware download

Firmware is installed into the device's Flash program memory via the USB link using Atmel's FLIP (Flexible In-system Programmer) PC utility. The latest version of FLIP software and user documentation can be downloaded from Atmel's website.

To start the device's USB bootloader, press and hold the ISP (In-System Programming) button, then press and release the reset button, then release the ISP button. The display should be blank. Plug in the USB cable to your PC. Start the FLIP application. Follow FLIP instructions from Atmel.

Host Command Interface (HCI)

The firmware incorporates a communications protocol intended primarily for passing data to and from a host PC via the USB link. The protocol is based on a simple command-line user interface format, hence the term 'host command interface' (HCI). Since the protocol uses printable ASCII characters, the HCI can be accessed using a terminal emulator (such as HyperTerminal) for human interaction. The HCI can be used interactively for setting clock options, parameters, etc. Advanced configuration options, not accessible via the 'local user interface' (control panel), can be set using the HCI with HyperTerminal, or a custom Windows GUI application (TBD).

The device USB port appears to the host PC as a virtual COM port. Windows application software can communicate with the clock through the standard Windows COM port API function calls (open, read, write, etc.).

A command string is composed of a 2-letter command mnemonic and a number of user-supplied arguments (parameters). Some commands have no arguments. A single space must be inserted between command line arguments, where there is more than one (including the two-letter command name). HCI input is generally not case-sensitive. There is no provision for command-line editing, but Escape or Ctrl-X will cancel the command line.

A brief command summary is given by the help command, 'HE'. Debug commands are provided to assist programmers developing their own

firmware code, and for those interested in the internal workings of the micro-controller and firmware.

For further details on the HCI, refer to the *User Guide* contained in the free archive download at [1].

Control panel operation — a brief overview

Figure 1 shows the standard layout of push-buttons along the top panel of the clock. The SNOOZE/CANCEL button is also located on the top panel. The buttons, LED display and encoder switch (knob) comprise the local user interface. Most clock operations and settings are accessible via the local user interface. Some of the more esoteric user options are adjustable only via the USB link.

Unlike most alarm clocks, the display on this one shows the day-of-the-week. When setting an alarm, the user can choose which day(s) of the week the alarm will be activated. Other LED enunciators show the four alarm events' on/off status (A1–A4) and the four time-switch outputs' on/off status (S1–S4). The control panel (**Figure 2**) allows manual override of the time-switch outputs.

Most of the push-buttons have more than one function. The function performed by a button press is dependent on the context of the operation. For example, in normal time-of-day display mode, pressing the [+] or [−] button will select a different item for display, e.g. seconds, year, date (month & day), etc. In a setting mode, however, the [+] and [−] buttons may be used to increment or decrement a numeric value, or to scroll through a list of items, e.g. days of the week.

In general, when in a setting mode, a flashing digit or LED enunciator indicates an item that can be changed by a button press or by turning the knob. The EXIT button quits a setting operation without making any changes.

Table 1 presents a summary of 'local user interface' (control panel) operations. The left column contains operations that can be performed from normal time-of-day display mode. For full details on operation, see the *User Guide* in the download package.

Next month's second and closing installment will discuss the schematics and the construction of the clock

100149

Table 1. Button functions	
Button press or other input	**Resulting action**
TIMER/CLOCK button hit	Countdown Timer (initial or remaining time, mm.ss) is displayed. From here, the Countdown Timer can be set and started.
ALARM CHECK button hit	Alarm Check mode is entered. The scheduled time of a selected alarm event (A1..A4) is shown. From here, the selected alarm (time and day-of-the-week) may be re-programmed, or enabled or disabled. The [+] and [−] buttons select the day-of-the-week for the alarm to be set. The ON/OFF button toggles the alarm status. Pressing the SNOOZE button while in Alarm Check mode allows the alarm sound effect and loudness to be changed.
TIME-SWITCH button hit	Time-Switch control mode is entered. A selected time-switch channel (S1–S4) may be switched on or off (temporarily over-riding the programmed time-switch schedule).
SET button held for >3 sec	Time-of-Day (TOD) setting mode is entered. The knob adjusts the hours, then if the SET button is pressed again, adjusts the minutes. Press SET again to commit to the change, or press EXIT to quit without any change.
MENU button held for >3 sec	Menu Setup mode is entered. From here, various user options and parameters can be adjusted. The [+] and [−] buttons scroll thru a list of menu items.
[+] or [−] button hit	The displayed item is changed from normal TOD to seconds or the date (year, month, day). Pressing [+] scrolls thru items in the sequence: Seconds (ss.cc), Year (20xx), Date (MM dd), then back to time-of-day. Pressing [−] reverses the sequence. The SET button can be pressed to set any of the displayed items.
SNOOZE/CANCEL button pressed	Alarm confirmation: The time of the next pending (enabled) alarm event, up to 48 hours in advance, is displayed. If the button is held down (> 2 sec), the alarm sound plays. Pressing SET while holding down the SNOOZE button allows the alarm sound type and loudness to be changed.
KNOB rotated (while normal time of day is displayed)	Display brightness is adjusted. Depending on selected options, the setting may be retained until next adjusted, or it may change again automatically.

CAN with BASCOM-AVR

By Mark Alberts
(Netherlands)

BASCOM-AVR [1], the popular Windows BASIC Compiler for AVR microcontrollers supports as standard quite a large number of peripherals. However, the CAN (controller area network) bus was an exception until recently. This bus was originally conceived as a communication system in vehicles, but these days CAN can be found in all areas of industry, from cars to rockets. There are different standards covering the details of the protocol, but they are essentially all very similar.

When Elektor presented the *Automotive CAN-troller* board (April 2009 [2]), two boards were quickly ordered. This board uses an AT90CAN32 processor. The big advantage of an AVR processor with a built-in CAN controller is that relatively little software is required and that the processor has plenty of time available to carry out other tasks.

Hardware
The circuit is quite complete in its implementation, but during testing a serial port is almost indispensable. Elektor sells handy USB/TTL interface cables (080213-71 [3]) which can be connected to TX1 and RX1. An additional advantage is that the USB bus can supply the necessary supply voltage. Of course, instead of a USB/TTL cable you could also use a trusty MAX232. However, in this case the board will have to be powered from a power supply adapter.

To establish communications we required a minimum of two boards. They are connected together via the CAN-bus (connector K2, see schematic). To program the controllers we use the Elektor mk.II programmer. Any other ISP-programmer with a 6-pin connector will also suffice. By default, the fuse byte of the microcontroller is set to use the internal oscillator, with the divide-by-8 divider activated. CAN requires a very stable clock, so a 12-MHz crystal is fitted on the board. The BASCOM $PROG looks as follows:

```
$prog &HFF , &HCF , &HD9 , &HFF
```

CAN demo program source code
```
On Canit Can_int                            ' define the CAN interrupt
Canreset                                    ' reset can controller
Canclearallmobs                             ' clear all message objects
Canbaud = 125000                            ' use 125 KB

Config Canbusmode = Enabled                 ' enabled,standby,listening
Config Canmob = 0 , Bitlen = 11 , Idtag = &H0120 , Idmask = &H0120 , Msgobject = Receive , Msglen = 1 ,
    Autoreply = Disabled                    'first mob  is used for receiving data
Config Canmob = 1 , Bitlen = 11 , Idtag = &H0120 , Msgobject = Disabled , Msglen = 1  ' this mob is used for sending data

Cangie = &B10110000                         ' CAN GENERAL INTERRUPT and TX and RX
Print #2 , "Start"

Do
  If Pinc <> Bdil Then                      ' if the switch changed
    Bdil = Pinc                             ' save the value
    Bok = Cansend(1 , Pinc)                 ' send one byte using MOB 1
    Print #2 , Bok                          ' should be 0 if it was send OK
  End If
Loop

Can_int:
_can_pageok = Canpage                       ' save can page because the main program can access the page too
  Cangetints                                ' read all the interrupts into variable _can_mobints
```

Software

In order to make working with CAN easier, BASCOM has been extended with a few special CAN-statements. The code is based on interrupt processing. This results in the lowest overhead on the processor.

A complete description of the CAN protocol is too much to have here, so we will limit ourselves to the commands. It is important to select the correct baud rate. This can be set with CANBAUD. Furthermore, you need to ensure that all CAN devices on the bus have the same baud rate. For these reasons multiple buses can often be found in cars, all with different baud rates.

CAN operates using message objects (MObs). The processor has 15 different MObs available. By setting the properties of a MOb we determine whether we are sending or receiving data. We can also determine whether we are sending 11-bit or 29-bit messages. Each MOb can be set completely independently of the other MObs.

For example:
Config Canmob = 0 , Bitlen = 29 , Msgobject = Receive , Msglen = 8 , Idtag = &H0000 , Idmask = &H0000 , Autoreply = Disabled

Here we configure MOb number 0 for 29-bit messages and we indicate that we want to receive a message. A maximum of 8 bytes of data can be sent or received. More data can be transmitted using several MObs.

Using the IDTAG we can filter which IDs we want to receive. With IDMASK we can also define a range. Using this method we can therefore wait for one specific instruction. All other IDs are ignored. To minimize the system load it is important to set these correctly. That is because we do not wish to interrupt the main process for messages that are not intended for us.

For example:
IDTAG=&H0123 , IDMASK=&H0123 : react only to ID &H0123.
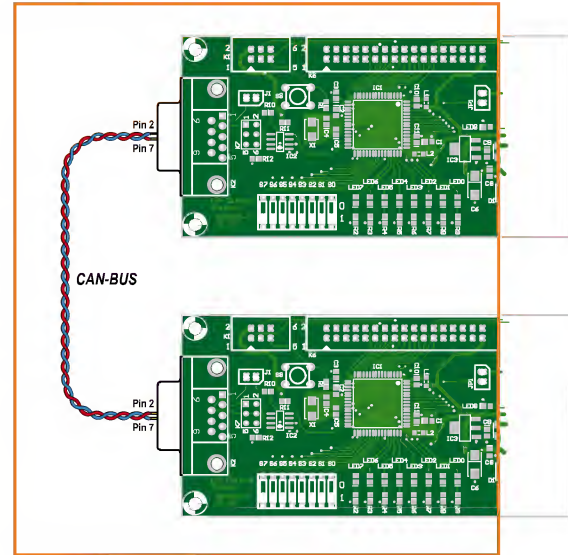

```
For _can_int_idx = 0 To 14                              ' for all message objects
    If _can_mobints._can_int_idx = 1 Then               ' if this message caused an interrupt

        Canselpage _can_int_idx                         ' select message object

        If Canstmob.5 = 1 Then                          ' we received a frame
            _canid = Canid()                            ' read the identifier
            Print #2 , Hex(_canid)

            Breceived = Canreceive(porta)               ' read the data and store in PORTA
            Print #2 , "Got : " ; Breceived ; " bytes"  ' show what we received
            Print #2 , Hex(porta)
            Config Canmob = -1 , Bitlen = 11 , Msgobject = Receive , Msglen = 1 , Autoreply =
                Disabled , Clearmob = No                ' reconfig with value -1 for the current MOB
        Elseif Canstmob.6 = 1 Then                      'transmission ready
            Config Canmob = -1 , Bitlen = 11 , Msgobject = Disabled , Msglen = 1 , Clearmob =
                No                         ' reconfig with value -1 for the current MOB and do not set ID and MASK
        End If
    End If
Next
Cansit1 = 0 : Cansit2 = 0 : Cangit = Cangit             ' clear interrupt flags
Canpage = _can_pageok
Return
```

•Projects

Overzicht commando's	
CANRESET	Reset CAN controller. Also happens during a hardware reset
CANCLEARALLMOBS	Erase all message objects
CANCLEARMOB	Erase a specific message object
CANBAUD	Set CAN baud rate
CONFIG CANBUSMODE	Set bus mode
CONFIG CANMOB	Set properties of the message object
CANGETINTS	Read the message interrupts
CANSELPAGE	Select a message object
CANID	Read CAN ID
CANRECEIVE	Read the data
CANSEND	Send the data

```
IDTAG=&H0123 , IDMASK=&H0120 : react
to ID &H0120 to &H0123
```

When sending a message, IDTAG is used to set the ID of the message. IDMASK has no function when sending a message.

Because several MObs can be active at the same time, it is also possible that they generate an interrupt simultaneously. The CAN interrupt service routine therefore checks each MOb whether it generated an interrupt. If this is the case, the data can be read using the CANRECEIVE function and the MOb is activated again. This is important: once a MOb has been used is it freed up. Without activating it again we would not receive any more messages. We do not, however, need to initialize all the properties again.

Program Flow CAN message object 0 and 1
MOb 0 is used to receive the data.
MOb 1 is used to send the value of the DIP switch when it changes.
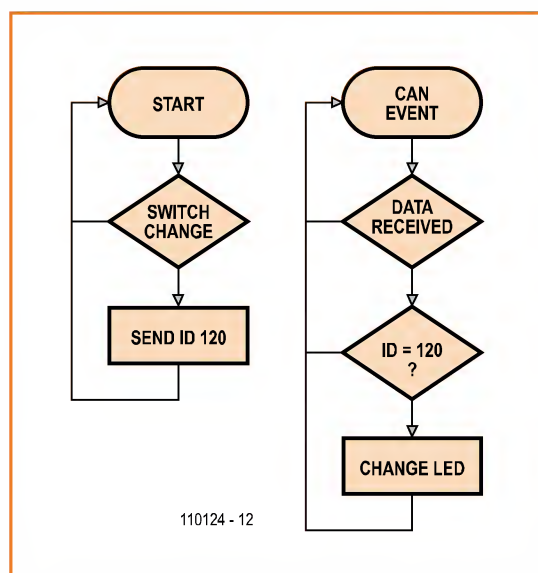
The *CANRECEIVE* function stores all the received data in the specified variable and returns the number of bytes received. For example: received=*CANRECEIVE*(portA).

It is important to keep any code in an interrupt service routine as short as possible. Here the data is only read and stored; a flag can be tested in the main program loop and the data processed. PRINT instructions are used in the example shown, but these and other slow commands normally need to be avoided in interrupt service routines.

The sending of data we do with *CANSEND*. We indicate which MOb needs to be used and we specify a variable and optionally the number of bytes that need to be sent.

For example:
```
ok= CANSEND(0 , ar(1) , 4)    'send 4
bytes from array AR to MOB 0
```

This function returns a '0' when the data has been correctly transmitted. The MOb is configured again in the interrupt-routine.

The Automotive CANtroller board has a DIP switch and 8 LEDs. We now create a program that has to be loaded into both controllers (can be downloaded free from [4], also see the flow chart). This program waits for an ID &H0120 containing one data byte. This data byte is then indicated on the LEDs. In addition the DIP switch is queried and this is sent with an ID &H0120. You will notice that these IDs are the same. This is possible because we cannot send messages within the same controller.

By changing the DIP switch on one board the LEDs on the other board will change accordingly, and *vice versa*.

(110124-I)

Internet Links

[1] www.mcselec.com
[2] www.elektor-magazine.com/080671
[3] www.elektor-magazine.com /080213
[4] www.elektor-magazine.com /110124
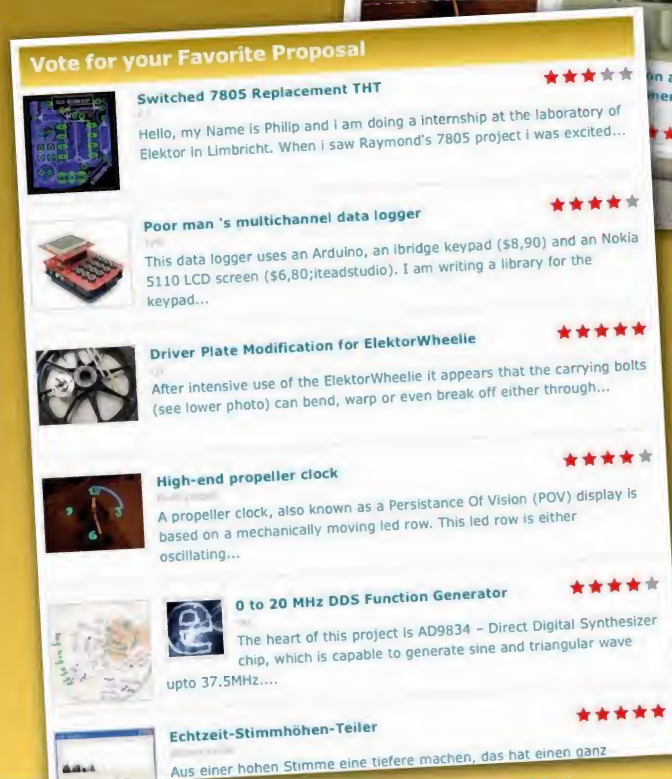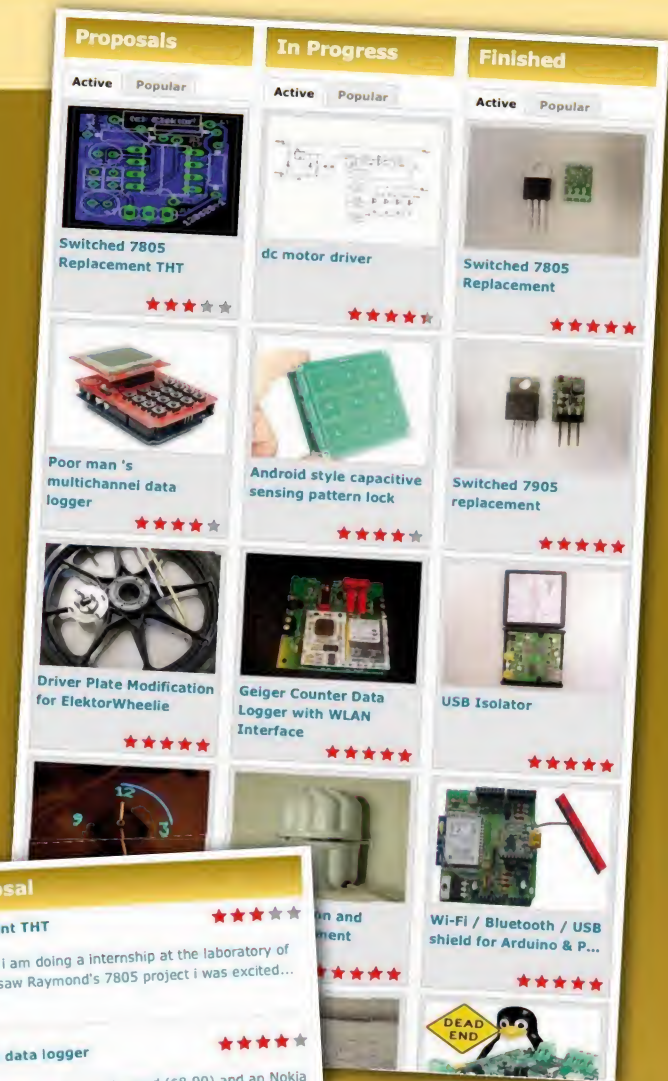
Kiddies Toothbrush Timer

By **Jere Manner**
(Finland)

A classic sandglass is sure to stir up feelings of authenticity and nostalgia. Although there's no electronics inside or insane noises produced, young children are instantly attracted to it, possibly as a result of the sand trickling down smoothly while they cannot stop it. Also, there's the innate perception that "time's running out". Still on elementary psychology, most children are unlikely to take their parents'

timing concerns seriously, like when it's bed time and the daily ritual of teeth brushing is due. Here's a playful, child-friendly way of teaching young'uns to time the duration of their daily teeth brushing round(s). C programmers and/or AVR nerds may want one too.

The timer presented here is designed to look like and mimic a classic sandglass and should need zero explaining to your kids, the operation being intuitive. If they can switch on the TV and the WiFi router, surely they can operate this timer too.

Press the button to start the timer. LEDs light in sequence to imitate sand grains trickling down, and (yes) a modest beep sounds every 30 seconds, as well as when "time's up". One LED equals 10 seconds.

The circuit is a typical example of a small microcontroller reducing component count considerably compared classic TTL, CMOS or NE555 real

COMPONENT LIST

Resistors
(SMD 0805)
R1 = 1kΩ 1%
R2 = 10kΩ 1%
R3,R4 = 75Ω 1%
R5–R16 = 120Ω 1%

Capacitors
C1,C2,C3 = 100nF 25V, SMD 0805

Semiconductors
D1–D24 = LED, red, SMD 0805, e.g. Kingbright KPHCM-2012SURCK; Farnell # 1686067
D25 = LED, green, SMD 0805, e.g. Kingbright KPHCM-2012CGCK; Farnell # 1686075
IC1 = ATtiny13A-SSH, SO-8 case, programmed, Elektor # 120368-41
IC2,IC3 = 74HC164D, SOIC-14 case
T1 = BC850C, SOT-23 case

Miscellaneous
BT1 = button cell holder and CR2032 battery; Farnell # 2064725
BZ1 = piezo buzzer, Farnell # 1192551
K1 = 6-pin (2x3) pinheader, 0.1'' lead pitch
S1 = pushbutton, side actuated, Farnell # 1761633
PCB # 120368-1, see www.elektor-magazine.com/120368
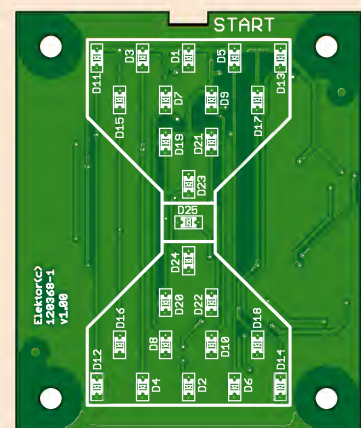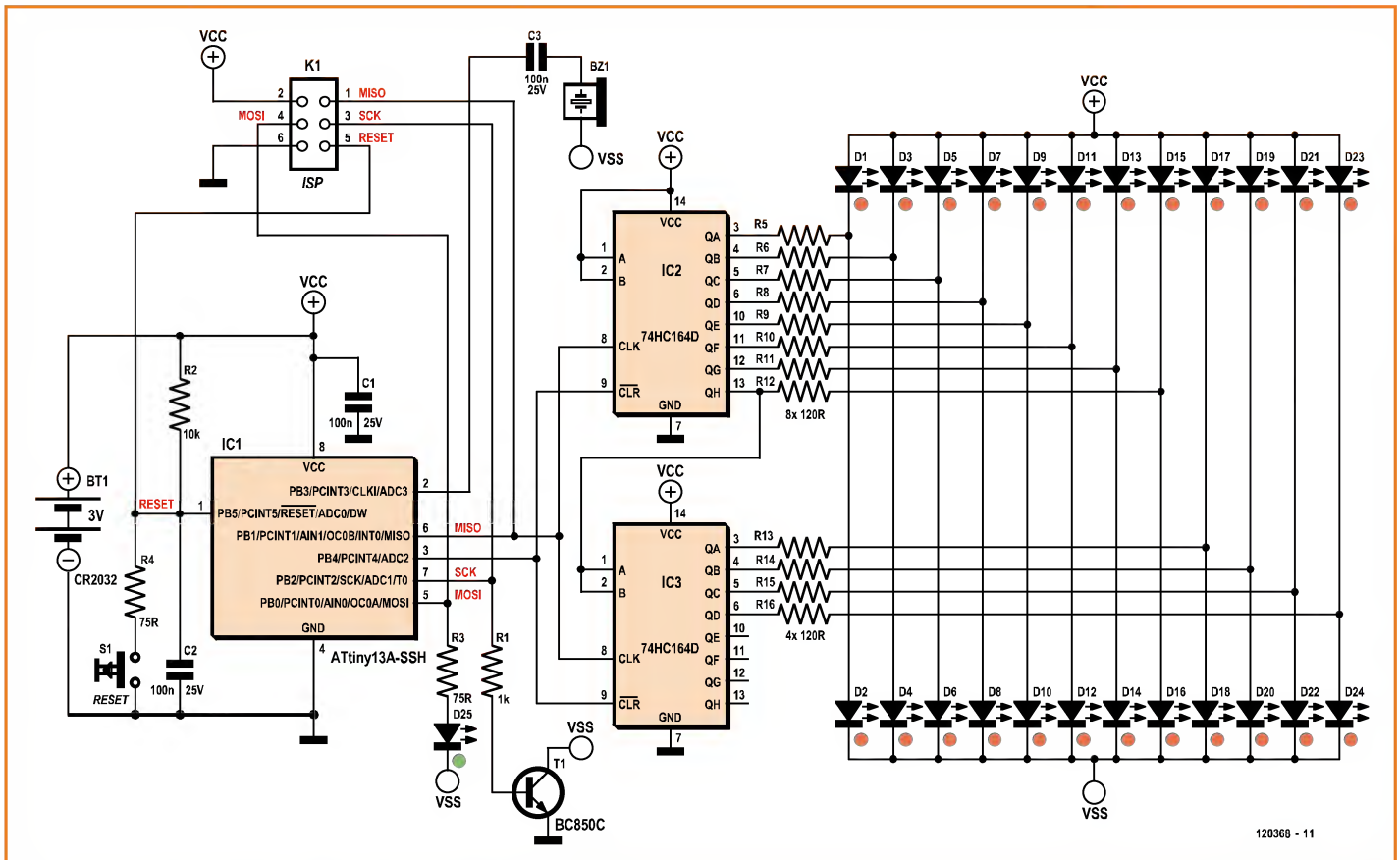
estate. An ATtiny13 micro is running firmware to control the timing for shift registers IC2 and IC3, as well as the buzzer. The 74HC164D shift registers drive an array of (SMD) LEDs though current limiting resistors R5–R16.

The crux of the project is in its appearance, i.e. the PCB design and the use of small parts to make a compact, lightweight device.

This project was post-engineered lightly in the Elektor Labs, with some minor but essential details to mention. Like R4 got added, and an ISP (in system programming) header to enable the software fans among you to burn their own chip, do mods, and so on. The software was revised at the C source code level, while a few components got changed due to local availability issues. Some concerns should be expressed about the battery capacity. The CR2032 button cell on the board will not last long due to the LED current, although the 'off state' current of the circuit is really small. Feel free to change the button cell to a more powerful variant, like dry AA cells.

The C source code file for the project is available free of charge at [1].

(120368)

Elektor Labs Tips & Tricks

Glue the battery holder to the board. We ripped the holder from the board while trying to the change the battery ☹.

Program the micro before soldering it on the board, or use an external power supply. The programming operation drains a CR2025 battery from 3.0 V down to 2.3 V → micro resets, goes to sleep → weird situations ☹.

A bolt or screw in a corner hole of the board makes a low-cost, industrial-look stand for the timer.

Internet Reference

[1] www.elektor-magazine.com/120368

Challenges for you

Design and produce a case for the device using a 3D printer.

Beef up the power source to cope with the LED current.

Add a tilt switch that starts the timer, just like turning a classic sandglass. Tip: review "Motorbike Alarm", Elektor # 120106.

Report back on www.elektor-labs.com to share your work.

13-IN-A-BOX
A handy multitester concept
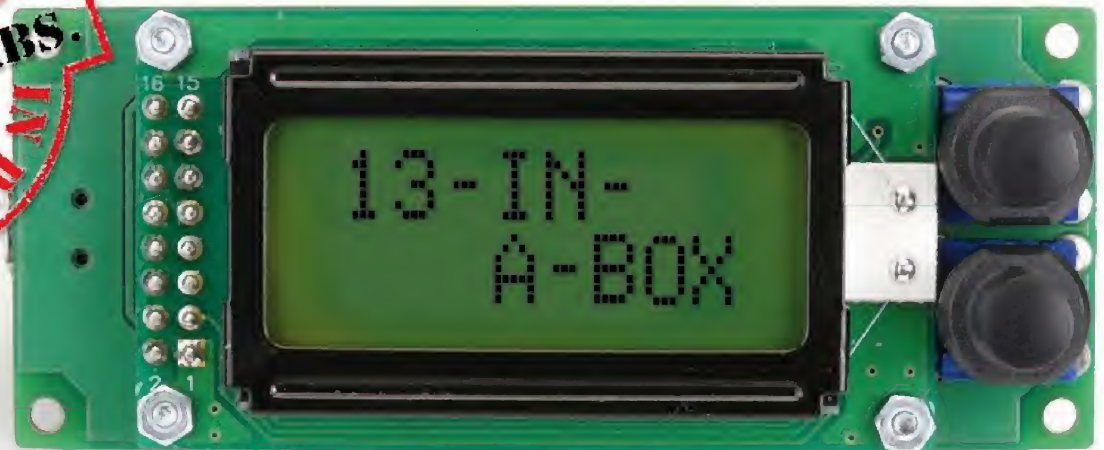
By **Thomas Ücok**
and **Jens Nickel**
(Germany)

When working on microcontroller projects there are lots of little tests you always need to do. Is that UART really transmitting anything? What's the voltage level on pin PC2? Have I configured the timer properly to measure the input frequency? All these questions and many more can be answered by a handy multitester developed by *Elektor* reader Thomas Ücok using our familiar Minimod module, and this forms the basis of an interesting Elektor project.

Many electronics engineers' benches live in a permanent state of organized chaos. When you want to make a quick test on some device, you usually have to make room for the oscilloscope, frequency generator and so on, and then switch on and set up the instruments. For most tests this is an unnecessarily complicated approach, as we shall see here. The author has used the Elektor Minimod module [1], which contains an AVR microcontroller and a display, to create a handy and easy-to-use multitester (**Figure 1**). A menu lets you choose from the following functions:

1. TTL logic level tester, 5 V or 3 V, high/low/open-circuit
2. Frequency measurement up to 7 MHz
3. CMOS logic level tester, 5 V, high/low/open-circuit
4. Analog voltage measurement, 0 V to 5 V
5. Pulse counter (rising edges)
6. Pulse counter (falling edges)
7. UART test receiver
8. UART test transmitter
9. Readout of ROM ID in a one-wire sensor
10. Pulsewidth measurement (low period)
11. Pulsewidth measurement (high period)
12. Squarewave generator at 1 kHz, 10 kHz, 100 kHz and 500 kHz
13. Pulsewidth measurement for servo control signals

Multitester: the next generation

Unfortunately the Minimod module is no longer available from Elektor. However, we were so impressed by Thomas Ücok's idea that we decided to develop the multitester as a new project rather than one dependent on the Minimod board.

The large picture shows an early prototype. The test results are shown on an alphanumeric LCD with two rows of eight characters. Power at 5 V is supplied over a USB connector, and so any PC or notebook, or any power supply with a USB output, can be used as a source. Two buttons are all that is needed to select the desired function and operate the device.

Any instrument that is to survive the rough and tumble of life on the bench will need a suitably robust enclosure. We have designed a dedicated enclosure for this project, which can be printed using a 3D printer. **Figure 2** shows a prototype.

The circuit

Figure 3 shows the current state of the circuit design, incorporating several of Thomas Ücok's clever ideas. The circuit is based around and ATmega328P, wired in the conventional fashion. It is clocked by a 16 MHz crystal and is powered at 5 V. An ISP connector allows the microcontroller to be programmed. The LCD module is driven in four-bit mode, and P1 allows its contrast to be adjusted. JP1 gives the option of activating the display's backlight.

The USB interface is mainly there to provide a power source for the circuit. However, it would be possible to use it to load new firmware into the microcontroller if the 'USBaspLoader' bootloader is installed [2]. This tool implements the whole USB protocol in software, and thus avoids the need for an external USB-to-serial converter in the circuit.

A 24C512 EEPROM is connected over the microcontroller's I²C bus: it can be used for storing configuration data.

The only unusual aspect of the circuit is the MAX4584 audio/video switch [3], which is also connected over the I²C bus. This switch is used to direct the test signal (for example from a test probe) to various inputs on the microcontroller. This is necessary because unfortunately there is no single pin on the device that can carry out all the different measurement and signal genera-

tion functions. Pin PC7.ADC7 is used to detect TTL and CMOS logic levels, to measure analog voltages and to count pulses, while pin PD4/T0 is used for frequency measurements. This pin is also used as a UART transmitter output.

The analog input is biased to 1 V by the voltage divider formed by R1, R2 and R3. This is not a valid TTL logic level, which allows an open-circuit input to be detected in TTL level mode. In 'analog' mode the device will always read 1 V if no voltage is applied to the input.

A timer input (T0) is used to measure frequencies. Pin PD4/T0 is, however, also used to drive the LCD. The trick is to ensure that there is no

Figure 1.
The multitester design by Thomas Ücok is based on the Elektor Minimod module.

Figure 2.
An enclosure for the project can be made using a 3D printer.
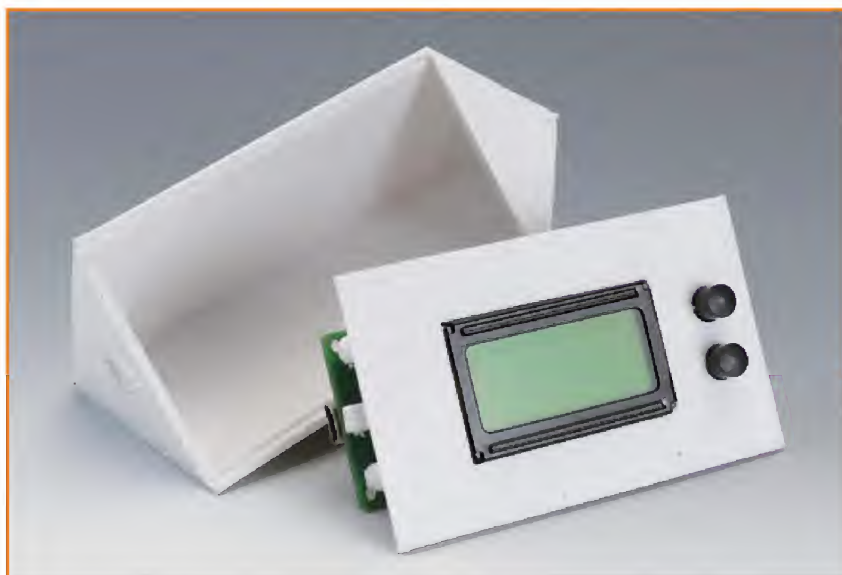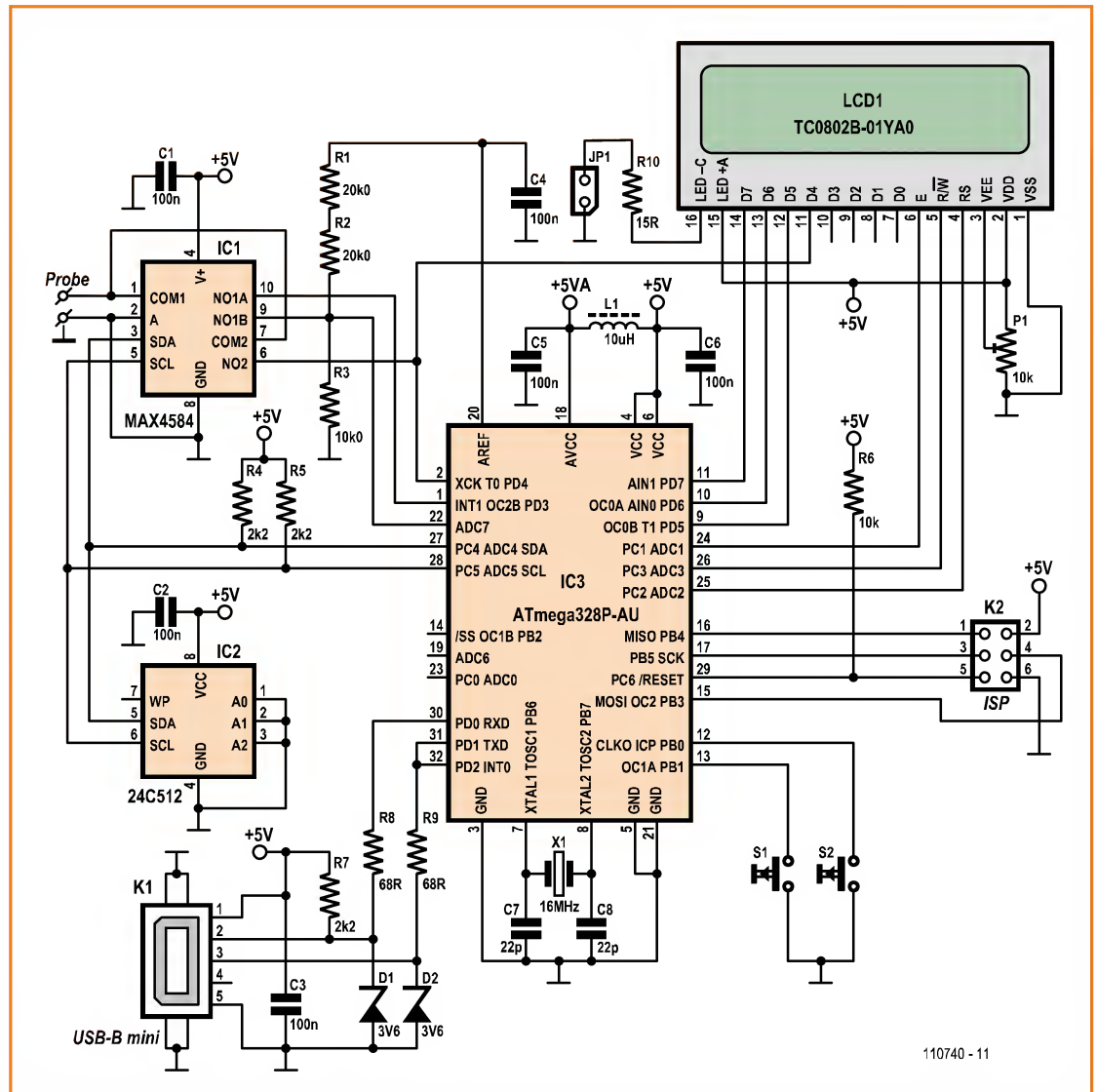
Figure 3.
The circuit is based around an ATmega328, which is pin-compatible with the popular ATmega88.

output to the LCD when making a measurement, so that the pin can be reconfigured during this period as a timer input.

Printed circuit board
The prototype printed circuit board is double-sided (see **Figure 4**). The buttons and display are mounted on the opposite side of the board from the other components. The display can be soldered in, although a plug-and-socket solution is preferable: solder PCB-mount sockets (for example from Harwin) to the display board and a two eight-pin headers to the multitester board. The header pins we used were thicker on one side than the other, and it is the thinner end that is fitted in the board. JP1 is mounted horizontally to

save space. The display is held in position using four M2 bolts and twelve nuts.
The probe and ground connections were made using 1.3 mm pins on the component side of the board, again bent horizontal so that the back of the module is kept relatively flat. The ISP connector only needs to be fitted if it will be used.

Software
The current prototype uses the author's software, which is written in BASCOM BASIC (from MCS). As with any project, the software offers many possibilities for extensions and enhancements.

You can keep up-to-date with the current state of the project on our Elektor Labs website [4].
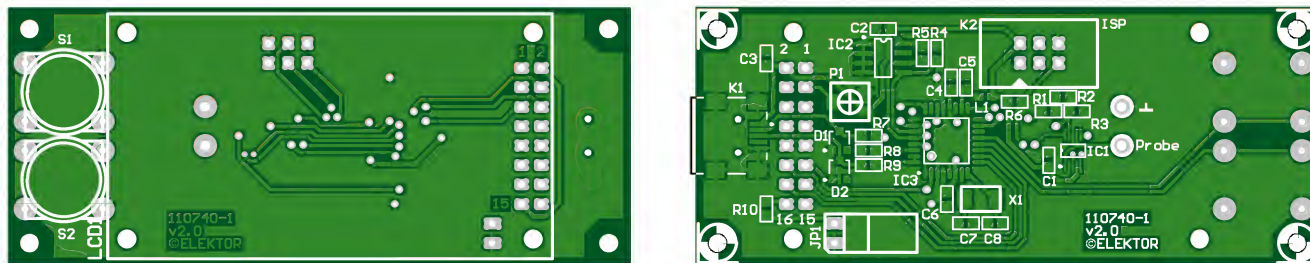
There you can also download the board layout, a parts list and of course the software. There is also a full description of how to use the various modes and features of the multitester's software. We of course welcome your comments and suggestions, as well as any extensions you make to the design!

(110740)

Internet Links

[1] www.elektor-magazine.com/090773

[2] www.obdev.at/products/vusb/usbasploader. html

[3] www.maximintegrated.com/datasheet/index. mvp/id/2080

[4] www.elektor-labs.com/9120802389

Figure 4.
The display and buttons are mounted on the other side of the board from the rest of the components.

Tropical or Arctic?
No Sweat!
This Temp. & RH Meter logs it all

By
**Manoel
Conde de Almeida**
(Brazil)

Digital thermometer or thermostat projects have been around for many years. This project, in order to be appealing in such an extensively explored subject, incorporates a number of interesting features never put together in one single circuit, some of which even only available on professional equipment.
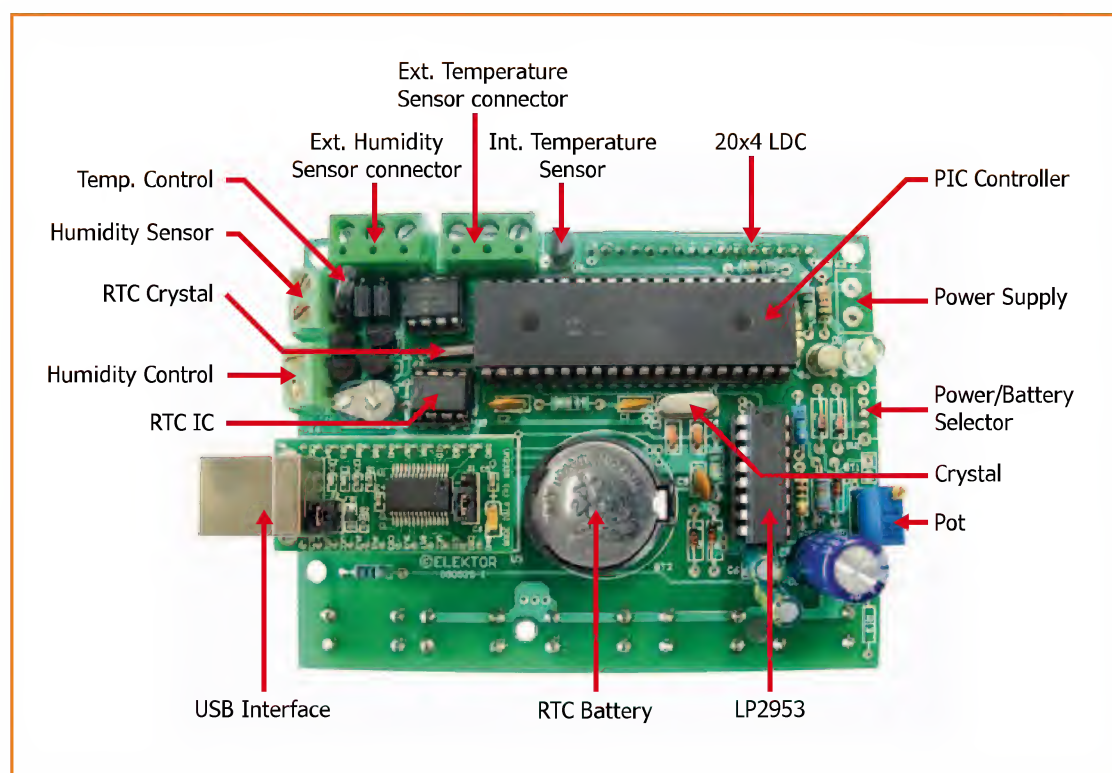
The main objective of this project is to build a circuit that is able to measure temperature and relative humidity. The final design incorporates a number of interesting features. As temperature and relative humidity measurements are taken, it keeps track of observed maximum and minimum levels. Each reading is also compared to user defined maximum and minimum limits. When these limits are exceeded, dedicated outputs are activated so they can be used to drive external circuits.

Temperature measurements are taken in either degrees Celsius or Fahrenheit. The setting of the maximum and minimum temperature and relative humidity limits is adjusted using a PC on which accompanying software is installed (available for free downloading [1]).

A digital clock keeps track of time, day, month and year, which is needed when the meter is collecting and storing measurements in its microcontroller-resident EEPROM. Temperature and humidity readings are intermittently taken at 7-second intervals. Up to 240 measurements (temperature and humidity) can be stored, collected in periods

that can be set from 1 to 99 minutes in incre-
ments of 1 minute.

Controlling the circuit

The interface consists of a 20x4 character LC
display and a set of seven switches (five push
buttons and two on/off switches) which let you
access all of the configurable parameters. The
meter can also be controlled by connecting it to
a PC using a USB interface. A PC application,
downloadable from [1], controls all the meter's
functions and extends its data collection func-
tionality by presenting measurements on a small
graphic display and providing the means to save

troller's PORT D and operates in 4-bit commu-
nication mode. Push button S1 turns on the LCD
backlight, and R13 and trimpot TP1 allow for
contrast adjustment.
U3 is a UM232R module based on the FT232R
chip from FTDI [3]. This module implements a
USB to serial UART interface. It's connected to
the microcontroller's PORT C through pins C.6
and C.7, which are configured as the Tx and Rx
lines of the microcontroller's internal UART. The
module is powered directly by the USB interface,
saving battery power for the rest of the circuit.
The RTC (Real Time Clock) circuit is built around
U8 (PCF8583 from NXP [4]). Trimmer C8 and
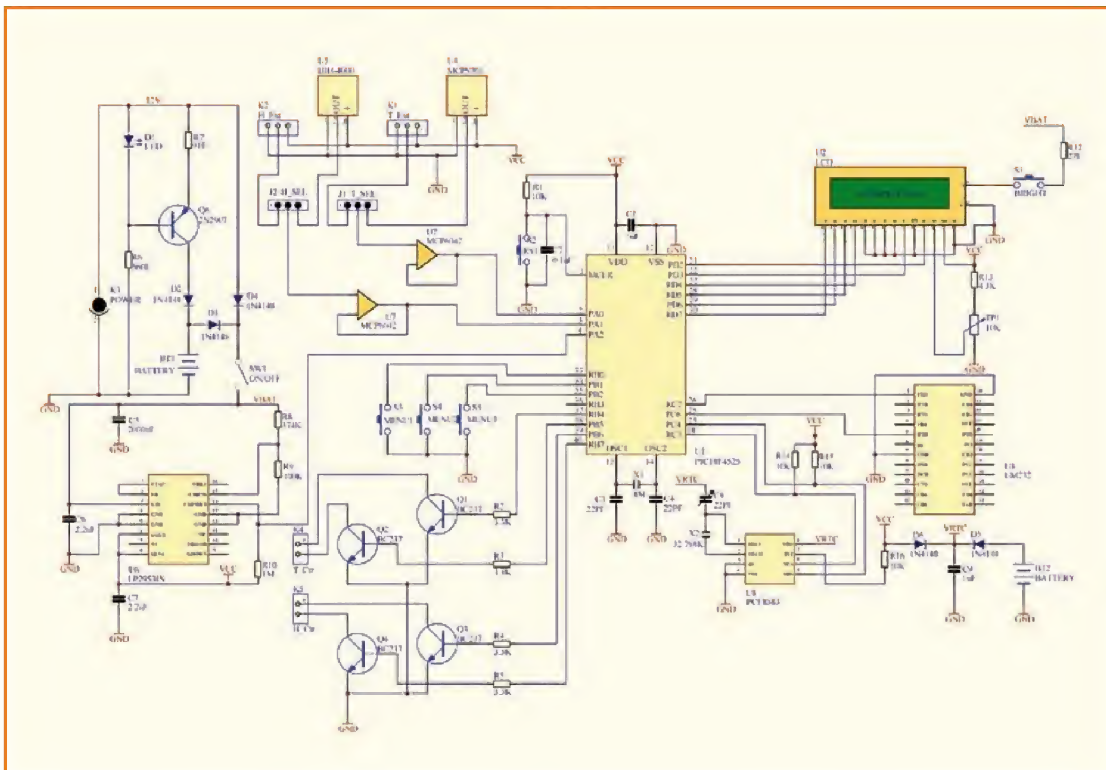
Figure 1.
This is the complete circuit.
Several sub circuits, like
the RTC and the battery
charging circuit, can be
spotted easily.

collected data in .csv files (comma separated
value) that can easily be imported in worksheets
for more complex processing.

Hardware highlights

The circuit, shown in **Figure 1**, is centered around
a PIC18F4525 microcontroller from Microchip [2].
It operates at a clock frequency of 8 MHz gen-
erated by the resonant circuit formed by quartz
crystal X1 and capacitors C3 and C4. The reset
circuit is formed by R1, C2 and push button S2.
The 20x4 LCD is connected to the microcon-

the 32.768 kHz crystal X2 form the resonant cir-
cuit. U8 communicates with the microcontroller
through an I^2C interface. Communication lines
SCL and SDA are connected to the microcon-
troller's PORT C pins C.18 and C.23 respectively.
Push buttons S3, S4, S5 provide access to the
function menu displayed in the LCD screen. Tran-
sistors Q1-Q4 and resistors R2-R5 build open-
collector output circuits driven by PORT B Pins
B.4 to B.7. These ports are activated whenever
the alarm function is enabled and temperature
and/or humidity maximum or minimum limits

Table 1. Alarm setting pin designation	
Port pin	**Alarm**
B.4	Temperature maximum limit
B.5	Temperature minimum limit
B.6	Humidity maximum limit
B.7	Humidity minimum limit

are exceeded. **Table 1** references each PORT B output pin to its respective alarm.

The circuit can be powered by a rechargeable 9-V NiMH battery that's recharged when the meter is connected to a 12-V DC power adapter. Transistor Q6, diode D1 and resistors R6 and R7 build a current source that provides a current of around 10 mA, which charges the battery whenever the AC power adapter is connected to K3. Diodes D2-D4 ensure proper power connection to the circuit. When a power adapter is connected, D2 and D4 pass current to the battery and circuit respectively, while D3 blocks the higher supply voltage, allowing the battery to be charged by the current source and the circuit to be powered by the adapter. When a power adapter is not connected, D2 and D4 block and the battery current passes D3, turning the current source off and allowing the circuit to be powered by the battery. Switch SW1 connects the power coming from the battery or the adapter to U6, an LP2953A low drop 5-V regulator from National Semiconductor. A voltage divider is implemented using resistors R8 and R9, providing a sample of the battery voltage. This voltage is compared to an internal voltage reference and whenever the battery voltage drops below 6 volts, the comparator's output (pin 14) will force microcontroller's PORT A pin A.2 to a low state. The firmware detects this status change and turns on the low battery indication in the meter's display.

Total power consumption is limited to 10 mA,

resulting in — depending on the capacity — around 15 hours of continuous operation with a fully charged battery. The RTC is powered by a back-up circuit that keeps it running even when the meter is turned off. The back-up circuit consists of BT2 (a 3-V lithium battery), diodes D5 and D6 and capacitor C9. When the external power is off, D6 is reversely polarized and BT2 supplies power to the RTC through D5. When external power is applied, the opposite situation exists. D6 forwards the current, D5 blocks it and the RTC is powered by V_{CC}.

Sensor specifics

Responsible for measuring the temperature is a Linear Active Thermistor™ IC from Microchip, the MCP9701. This 3-pin IC generates a voltage level linearly proportional to the temperature following the formula:

$$V_t = 0.5 + 0.01T,$$

where V_t is the output voltage in Volts for a given temperature T in Celsius.

The thermometer output is buffered by U7A, one of the two operational amplifiers available in the MCP6042 (low power Microchip operational amplifier). Both opamps are configured as unity gain buffers.

The relative humidity measurement is carried out by a Honeywell [5] HIH-4000 analog hygrometer. This sensor also provides a voltage linearly proportional to the humidity, given by the following formula:

$$V_h = 0.7617 + 0.0297H$$

where V_h is the output voltage in volts for a given relative humidity H (%rh).

The hygrometer output is buffered by U7B. Both the op amp outputs are connected to PORT A pins A.0 and A.1 of the microcontroller, which are configured as 10-bit A/D-converters.

Connectors K1 and K2 allow for the connection of external temperature and humidity probes (provided these use the same sensors as described above). J1 and J2 are jumpers used to switch between the internal and external sensors. The sensor equation parameters can be fine tuned using the PC application program.

Firmware facts

In order to keep the hardware as straightforward

as possible, the firmware takes over the bulk of handling the complexity of the meter's functions. It's developed using MikroBasic v7.2 from Mikro-elektronika [6]. The flowchart in **Figure 2** briefly describes the program's main routine, though a large portion of the program resides in subroutines which are not described here.

The main program starts with a series of initialization routines that include the microcontroller's internal register configuration, LCD module initialization, EEPROM initialization with the standard meter configuration values (if not programmed yet), program variables initialization, TIMER0 reset and TIMER0 interrupt enabling.

After the initialization routines the program will operate in an endless loop (the 'On' loop). In this loop the routine steps through the following states:

- check the battery state and turn on or off the Low Bat indicator appropriately.
- check whether the time has changed and update the main screen with the new time and calendar values.
- check if there is new measurement data available and update screens appropriately.
- check menu keys to identify user requests for screen or function changes and redirect the program flow accordingly.
- record data if data collection is enabled.
- check whether a PC is connected and execute any received command.

Although not described in the flowchart, it is important to note that all time dependent activities executed by the program are based on TIMER0 overflow interrupts. Those interested in understanding the complete program structure are welcome to study the complete program's source code, which is part of the project's documentation at [7].

PC program

The PC application was developed in Visual Basic 2008 Express Edition, available for free from Microsoft. Its operation is briefly described in this section. For more details, please refer to
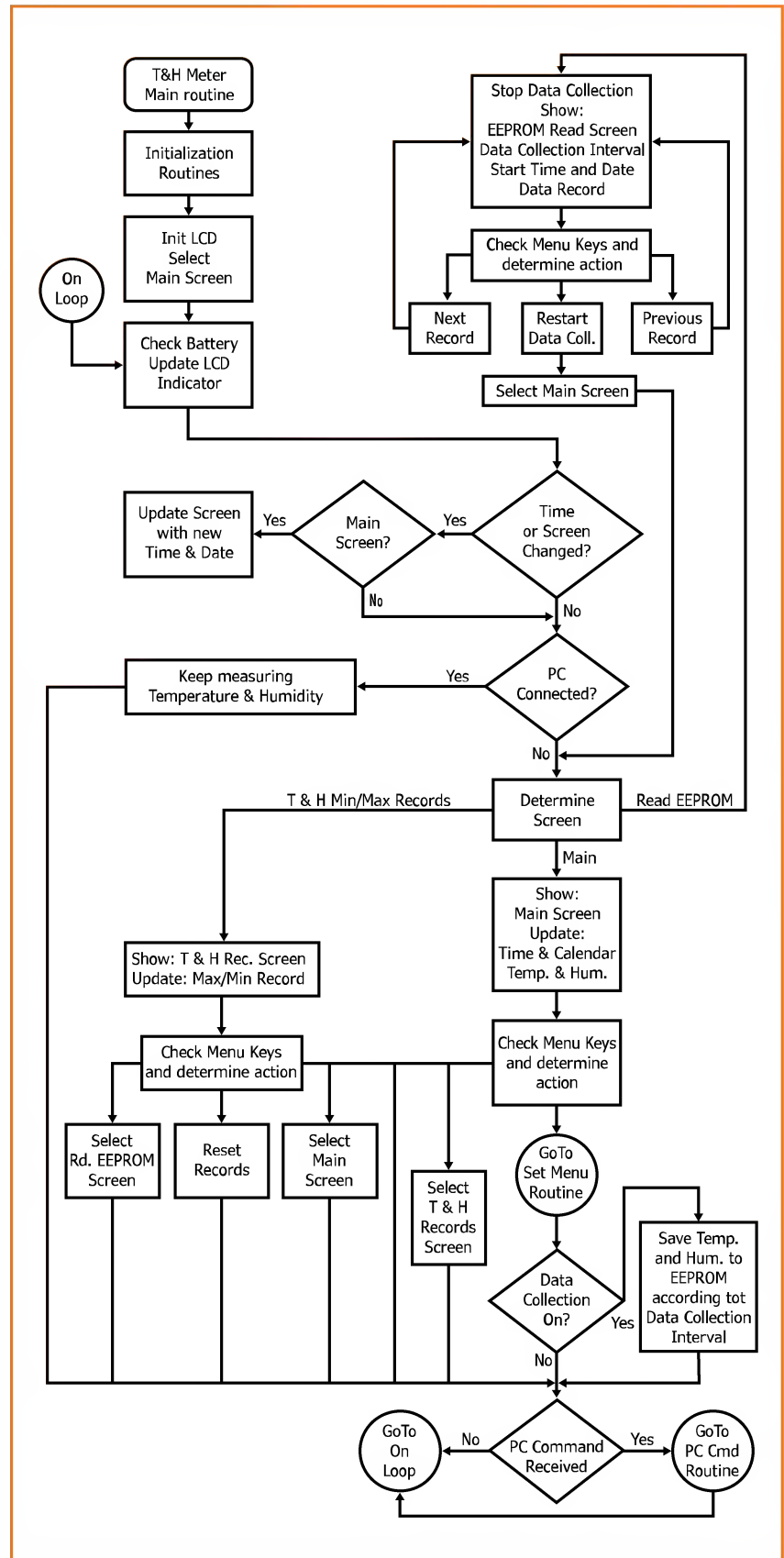
Figure 2.
The flow diagram displays the main firmware routines executed by the microcontroller.
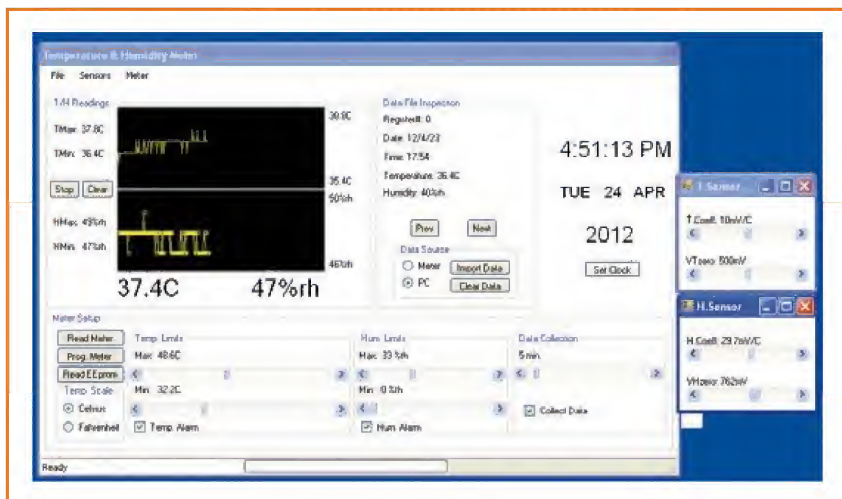
Figure 3.
The Graphical User Interface allows easy access to the various settings of the Temperature & Humidity Meter.

the online document at [1] which has a more elaborate description of the software installation and operation.

In order to establish communication with the meter over the USB interface, the software makes use of the free D2XX driver, available at the FTDI website [8]. **Figure 3** shows the PC program's main window. The *File* menu provides access to functions that allow the user to handle (clear, copy) the .csv data collection files. The *Sensors* menu holds functions that allow fine adjustment of the temperature and humidity sensors parameters. The meter is connected to or disconnected from the PC application via the *Meter* menu.
The *T/H Readings* frame shows the temperature and humidity readings in a line graph format. On the left are the highest and lowest recorded measurements and at the bottom, in bigger and bold text, the latest measurement taken. The Meter's clock can be set and synchronized to the PC clock by clicking the *Set Clock* button.

In the *Data File Inspection* frame the contents of the .csv data collection files can be read. Data is imported into the program's file inspector by selecting the desired source and clicking the *Import* button. The *Prev* and *Next* buttons are used to move through the data records. The *Clear Data* button clears the file inspector's data arrays, but will leave the original data files untouched. The *Meter Setup* frame groups all controls used to access the Meter's and program's operating parameters. Every time the program is launched and a connection to the Meter is established, its configuration is automatically downloaded. All

controls are set in accordance to the imported setup data. Modifications made to the current setup will immediately affect the application operation. In order to make the changes effective in the Meter, except for temperature scale changes, the new setup must be uploaded by clicking the *Prog. Meter* button.

The application monitors the *Temp.* and *Hum. limits* and indicates whether they have been exceeded by marking the respective measurement point in the graphic monitor. The check boxes *Temp. Alarm* and *Hum. Alarm* are used to turn on the respective monitoring functions. The *Data Collection* slider determines the time interval for data collection, which is switched on with the check box. Turning data collection on for the application results in capturing temperature and relative humidity measurements in the specified intervals and save them in the pc_cd.csv file. For the Meter it results in capturing measurements and storing them in the EEPROM. Csv-files are easily imported in PC software like Microsoft Office Excel or the Sun OpenOffice Calc which can be used for further and more complex data processing (reports, calculations, graphs, et cetera).

Like with every project described in Elektor, the firmware for this project as well as the PC program can be downloaded from this project's webpage [1]. The Component List is available online, as are the PCB layout and additional information on installing and operating the software. And please feel free to leave your improvements and/or comments on the Elektor Projects site [7].

(090925)

Internet Links

[1] www.elektor-magazine.com/090925

[2] www.microchip.com

[3] www.ftdichip.com

[4] www.nxp.com

[5] sensing.honeywell.com

[6] www.mikroe.com/mikrobasic

[7] www.elektor-labs.com/9120302013

[8] www.ftdichip.com/FTDrivers.htm

2-wire Interface version 2.0
Now with less current!

By Klaus Jürgen
Thiesler (Germany)

Reducing the number of interconnections in any system improves reliability. In June 2012 we published an idea by this author which uses just two wires instead of the more usual three to interface a switch and indicator light [1]. The author has further refined the circuit to use less current and to make it compatible with controllers operating at a supply voltage of just 2.1 V. That should be reason enough to give this circuit a second look. Just to recap: Usually three leads are necessary to wire a switch with indicator light to a microcontroller; this circuit gets away with just two, one of which is a common earth.

How it's done
The trick used here is to wire the indicator LED in parallel to the switch contacts and pass a low level of 'sense' current through the LED. The sense current is just a few microamps; not high enough to cause the LED to light up but sufficient to generate a forward conduction voltage of around 1.1 V to 1.7 V across the LED. A current of a few milliamps is necessary before the LED glow becomes visible and this also produces a higher forward voltage drop. When the LED is off a voltage of between 1.1 and 1.7 V can be measured across the LED which drops to zero when the push button is pressed.

The circuit
Unlike the circuit shown in the June issue this one uses two unusual PNP transistors. They have been chosen for their low value of V_{CEsat}. Despite its relative unfamiliarity the transistor is low-cost and stocked by Newark in the States or Farnell in Europe. R3 and R5 provide the low-level 'sense' current (2.5 µA approximately) through the LED. The microcontroller output pin Port.1 needs to be pulled 'low' in order to turn the indicator LED on. In this condition T1 is acting as a constant current source giving a voltage drop of 0.3 V across R1. With R1 = 27 Ω this limits the current through the LED to around 10 mA.
The values of R3 and R5 have been chosen so that transistor T1 starts conducting when the pins of K2 are shorted together. The level of

input Port.2 then goes 'high' indicating that S1 has been pressed. A low-pass filter formed by R5 and C1 suppresses any interference which may be induced in the interconnecting leads between K1 and K2. The circuit's quiescent current is effectively equal to the LED current. With S1 not pressed this amounts to just 2.5 µA.

Further tweaks
The circuit can also be adapted for applications where the supply rail is at a different level to the 2.1 V used here. Microcontrollers now commonly run on 3.3 V and less. With V_{CC} = 1.8 V

120071 - 11
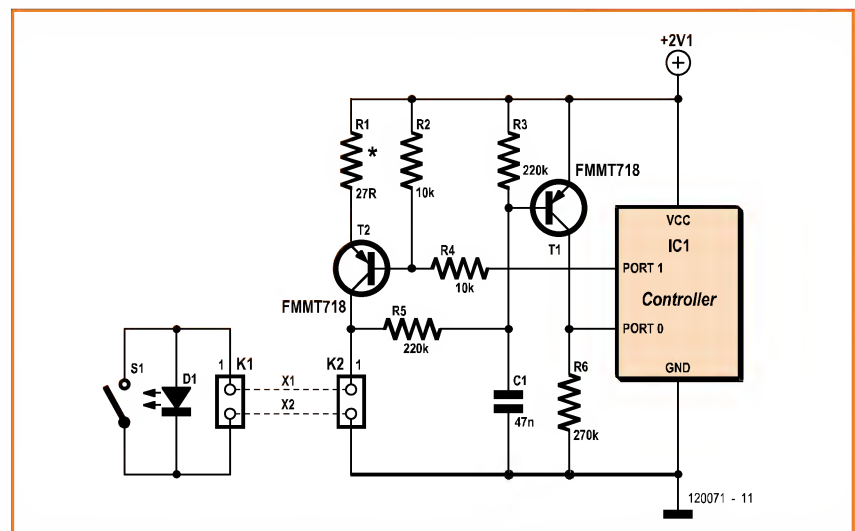
use a value of 8.2 kΩ for R4 and 160 kΩ for R5. At 3.3 V these values should be increased so that R4 = 22 kΩ and R5 = 470 kΩ.
With V_{CC} ≤ 2.1 V choose a red indicator LED. Green and yellow LEDs have a slightly higher forward conduction threshold and will only work above this supply level. Blue or white LEDs have a higher conduction threshold and are not suitable for use here. Higher efficiency LEDs use less current and will work with a higher value of R1; choose 180 Ω to give 2 mA through the LED.

(120071)

[1] www.elektor-magazine.com/110572

Simple On-bike Power Supply

5 V from muscle power

By Philip Jaschewski
(Elektor Labs trainee)

Thirteen years into the new millennium, people no longer take printed maps on cycling trips. A smartphone suffices! However, with power for recharging the phone seldom close to hand, a 5 V supply on board the bike would be extremely useful. This article shows just how easy this is to provide.

The day is close when your bicycle will provide an energy source for all those electronic gadgets you take with you. This prospect is entirely achievable, now that virtually all portable electronic devices are happy using 5 V, meaning a single bicycle power supply should be all you need for this diversity of devices. Its sole task is to convert the output from the dynamo into a 5 V direct current voltage. That's our objective — now comes how we translate this into a practical solution.

The standard bicycle dynamo delivers a nominal 6 volts with an output of 3 watts. Two implementations dominate (bit not in all countries): classic outboard dynamos operating from the wheel with a friction drive, and internal hub dynamos. Both types deliver AC voltages at some tens to some hundreds of Hz. The AC voltage needs to be converted to DC and then into a stable 5 V. Achieving these requirements calls for no magical powers and not even one single microcontroller.

As the circuit in **Figure 1** shows, the cycle power supply comprises a voltage regulator of conventional design plus a bridge rectifier. Other features relate to the characteristics of dynamos: under no-load or lightly loaded conditions the voltage rises in remarkably linear fashion with the number of RPM. At moderate speeds we have already more than 6 VAC reaching the bridge rectifier. Racing cyclists will produce well above 12 VAC. This is alternating current with a waveform fairly close to a sine wave, meaning that capacitor C1 charges to peak voltage under modest load conditions. IC1 can handle 25 V on its input without worry but for safety's sake we include a zener diode (D1) to prevent excessive voltage rise.

A double-pole switch (S1) is provided at the entry to the circuit, which we use to switch the supply between the cycle power supply and the front and rear lights. It's not a good idea to feed both loads at the same time, as even if you are convinced you can squeeze more than

Technical data

- Switchable between lights and power supply
- Output voltage: 5 V
- Output current: 0.5 A min.
- Designed for powering/recharging portable devices

the nominal 3 W out of the dynamo, you can never get twice that amount. So either lights *or* power supply.

To ensure reasonable efficiency, we use a so-called low drop-out (LDO) regulator for IC1 instead of a 7805, meaning that the voltage regulator will work effectively even when the differential between input and output voltages is small. 'Normal' 7805 versions require at least 8 V on the input. Improved variants can make do with 7 V. Other ICs such as the pin-compatible LM2931T manage down to less than 5.6 V. The voltage drop across the bridge rectifier is something we can also reduce a little by ditching conventional silicon diodes for Schottky types such as SB140 or 1N5817.

The layout files can be downloaded from the Elektor web page [1] for this article. You can also order a professional ready-drilled printed circuit board there.

The components used have no complicated connections, so populating the PCB will not present any problems. **Figure 2** shows a 3-D model of the completed board. A small 6 V transformer is ideal for testing the finished assembly. On the bicycle the project should be housed in a waterproof, insulated plastic casing, as electronics and water are definitely not close friends.

Figure 1.
The circuit of the cycle power supply is remarkably simple.
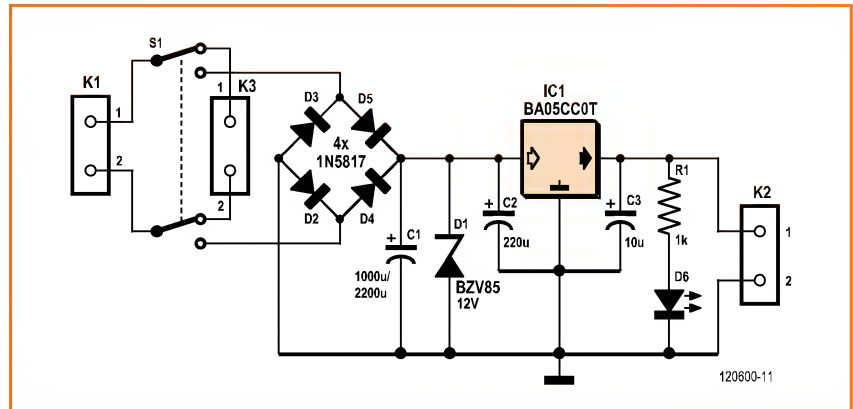
Something to remember is that many cycles and a fair number of dynamos use the exterior or frame of the bicycle as a common return. Under these circumstances the dynamo may have only one connecting point. In this case the second connection from the PCB (e.g. K1-2) goes to the frame. The ground connection of output K2-2 then must not come into contact with the metallic parts of the bicycle under any circumstances — nor even via any gadget connected to the power supply.

(120600)

Internet links

[1] www.elektor-magazine.com/120600
[2] http://eagleup.wordpress.com

COMPONENT LIST

Resistors
R1 = 1kΩ

Capacitors
C1 = 1000µF 35V, 5mm pitch, 10 mm diam.
C2 = 100µF 35V, 2.5mm pitch, 6 mm diam.
C3 = 10µF 16V, 2.5mm pitch, 5mm diam.

Semiconductors
D1 = 24V 1W zener diode
D2–D5 = 1N5817 or SB140*
IC1 = BA05CC0T (LDO, 1A, TO220 case)*
LED1 = LED, green, 5mm

Miscellaneous
K1,K2,K3 = 2-way PCB screw terminal block, 5mm pitch
S1 = switch, DPDT, 24V 1A
PCB # 120600-1
Suitable plastic housing and connection clamps

* see text
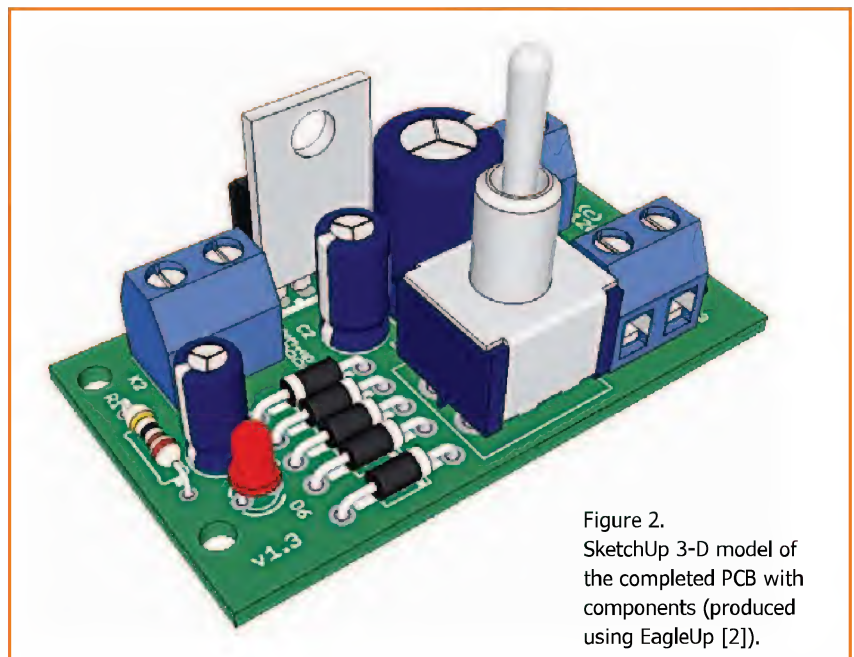
Figure 2.
SketchUp 3-D model of the completed PCB with components (produced using EagleUp [2]).

Capacitive Proximity Switch

Switching without wear

By Martin Jepkens,
Dr. Thomas Scherer
& Daniel Wunsch
(Germany)

People building their own self-balancing electric scooters face the problem of determining unconditionally whether the rider is standing on the platform or not. Mechanical solutions have not provided convincing answers. A purely electronic 'rider detector' exists already in the form of a capacitive proximity switch. Naturally this can also be adapted for other potential applications.

The three authors are building their own personal transporter vehicles, either as a variant of the ElektorWheelie or else as self-balancing two-wheelers along the lines of the Segway scooter. To prevent scooters from running away without their riders and ensure they stop when their riders alight (or fall off), we need to detect unconditionally whether the rider is still aboard the platform or the scooter is suddenly flying solo. For this function the Segway relies on mechanical switches operated by the weight of the rider.

Similar solutions are favored world-wide by homebrew constructors, although replicating the quality and reliability of the industrially-made original is much harder to achieve in the home workshop. Off-the-shelf switches are either too expensive, not waterproof, mechanically unstable or simply clunky in appearance. From this background emerged the idea of detecting the rider capacitively.

Capacitive methods

The first test attempted to detect the deflection of the platform surface and operate a micro-switch when the weight of the rider caused sufficient deflection. Good microswitches require only around 1.5 mm travel to operate. But the need to mount the microswitches with adequate access for mechanical adjustment made the task more problematic than desired.

A metal plate that flexes does produce a superb variable capacitor, if one plate of the capacitor is fixed rigidly. All you need do is measure the capacitance that varies when a rider stands on the upper plate and the job's done. Any adjustments necessary can be carried out both electronically and also automatically (if you use a small microcontroller to measure the capacitance).

So far, so easy. As so often happens, however, the devil lies in the detail. As we all know, there are several ways of measuring capacitance. We can employ the capacitance existing between two plates as the frequency-determining component of an astable multivibrator. We could also capture the charging current of the plate capacitor. Or we could use a fully tried and tested technique that was somewhat more involved and patented, to deliver secure and dependable results.

The microcontroller-astable multivibrator in **Figure 1** is a very simple affair: the capacitance to be measured C_x is charged and discharged between the output pin A and resistor R. Since the input (pin B) of commonly used microcontrollers has hysteresis like a Schmitt trigger, the necessary firmware is very straightforward. If

the voltage on input pin B is 'low', output pin A goes 'high' and if input pin B is 'high', then output pin A goes 'low' again. In this way we get a squarewave signal at pin B with a frequency determined by C_x. Now all we need is to use a timer to measure the frequency or switching period, and then produce a value for the capacitance. When someone mounts the platform, the distance between the 'footboard' and the fixed inner panel is reduced, increasing the capacitance and lowering the frequency. This can be established experimentally.

True to the KISS principle ('keep it short and simple'), we experimented first with the microcontroller version of an astable multivibrator, measuring the frequency/time period that resulted. Unfortunately the theory behind this was not crystal-clear: multiple experiments led to the conclusion that things were not that simple. The parallel plates serving as sensor below the footboard had a surface area of approx. 15 cm^2, separated by 1 mm. For whatever cause, they produced only a few tens of pF when deflected. To keep the frequency generated sufficiently low, R needed to have a value in the MΩ region. The arrangement no longer differentiated an open input properly and reacted to every disturbance. In practice the circuit not only lacked reliability but no longer delivered adequate sensitivity.

Fortunately we didn't have to reinvent the wheel from scratch. A firm called Quantum Research Group had patented a technique called QTouch that functioned very well. A couple of years ago Atmel acquired this firm together with QTouch and applied the technique to its own microcontrollers, which supported the idea of adopting this proven technology.

QTouch

The 'Q' in trade names doesn't always stand for 'quantum'; Q is also the symbol for electrical charge. This is precisely the case for the QTouch technique, in which the electrical charge from the sensor pad C_x is transferred to a larger capacitor. Surprisingly little is needed to achieve this. In principle you have only to replace the resistor R of Figure 1 with capacitor C_L, which takes you straight to the schematic in **Figure 2**.

The method does not take long to explain: the comparatively large capacitor C_L is charged cyclically via the relatively small capacitor C_x, until C_L is 'full'. At the same time we count the number
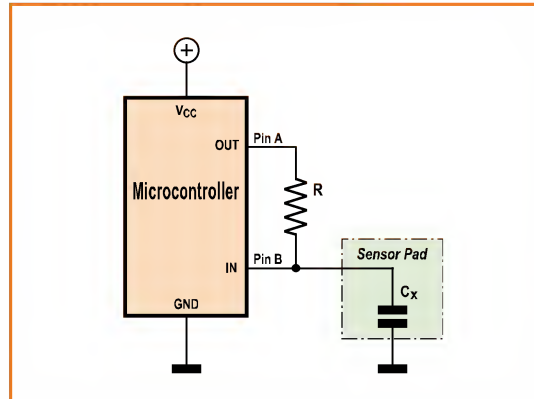
Figure 1.
Block diagram of an astable multivibrator using a microcontroller, in which the frequency is influenced by sensor capacity C_x.
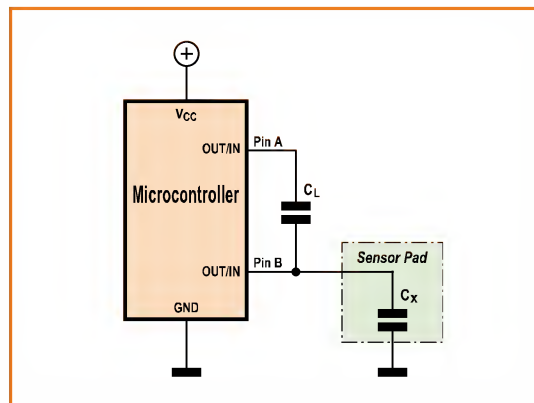
Figure 2.
Block diagram of the QTouch technique using a microcontroller. In contrast to Figure 1, R is replaced by C_L and the input is turned periodically into an output.

of cycles required. Finally C_L is discharged and the whole business begins afresh. For full details take a look at the **Pseudocode** inset. The only precondition as far as the microcontroller is concerned is that it must be able to toggle the use

QTouch Pseudocode

C_L and Cx are discharged at the final stage (line 2).

```
 1  Pin A and Pin B = out 'Pins as output
 2  Pin A and Pin B = low 'Discharge CL and Cx
 3  Cycles = 0 'Erase cycle counter
 4  Pin B = in 'Pin B as input
 5  Pin A = high 'CL is charged to an extent via Cx
 6  Cycles = Cycles + 1 'Increment cycle counter
 7  If Pin B = low then goto 1 'CL is fully charged, new start
 8  Pin A = in 'Pin A becomes high impedance
 9  Pin B = out  'Pin B as output
10  Pin B = low  'Cx discharged
11  goto 4 'Next cycle
```

If C_L is charged (pin B = low) in line 7, the variable 'Cycles' reaches the value corresponding to the capacitance of Cx.

Figure 3.
Circuitry for capacitive
proximity switches using the
QTouch technique.

handles 100 mA without difficulty. D1 protects the reset pin against over-voltage. To enable the circuit to be installed as a complete standalone module (for example as a foot switch for the ElektorWheelie), IC2 is also included on the PCB for stabilizing the power supply. D3 protects against reverse polarity. The supply voltage can therefore range between 7 and 14 V.

Figure 4 shows the finished prototype. The sensitive section around C1 is surrounded by a grounded area. The pad serving as second capacitor plate of the footboard is located on the rear of the PCB (see **Figure 5**). Why? The rear side is positioned opposite the footboard. To avoid mechanical disruption, the components are placed on the other side. For this reason we use an SMD component for the six-way pin header K4 provided for programming the controller. The shape of the pad is not critical but its total surface area needs to be maintained fairly closely. After component placement, programming and testing, the PCB should be sprayed with transparent lacquer if it is to be installed on a vehicle. Make sure K4 is masked off when you do this! The lacquer protects against problems caused by condensing humidity etc. After this glue a thin layer of rubber (around 0.6 mm thickness) onto the rear side of the PCB, to serve as a kind of buffer in case the footboard deflects too far. Glue somewhat thicker sheet rubber (say 1 mm thickness) onto the ba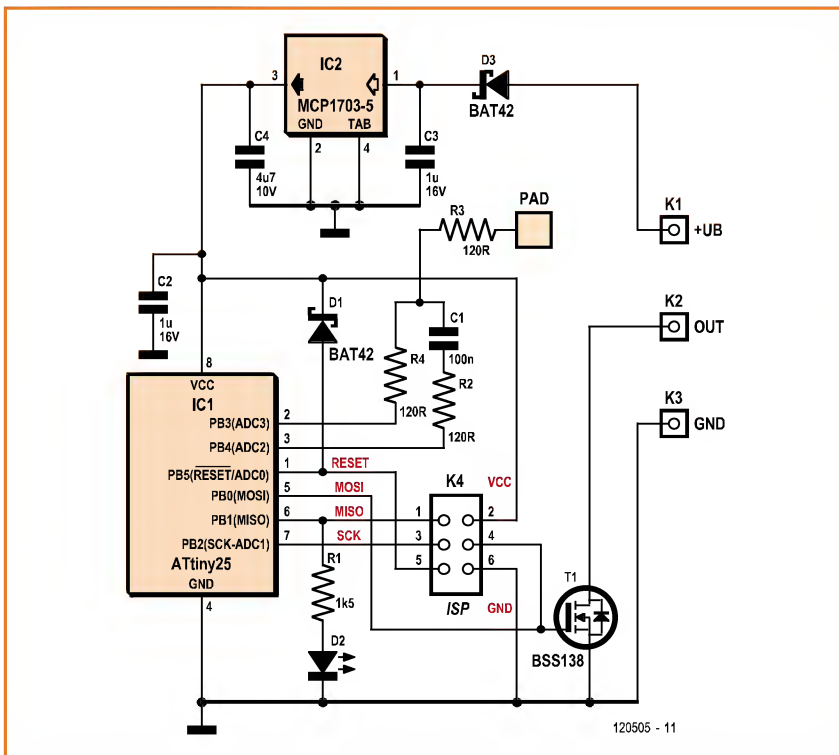re area of the component side. The upper side protected in this way along with the rubber layer should have a matching piece of metal fixed firmly as a sub-layer, so that the PCB can serve as a virtual end stop for the footboard.

of its pins between input and output. Since the capacity of C_L is significantly larger than that of C_x, the impedance on pin B (when measuring levels) drops notably lower than in Figure 1. This makes the QTouch technique fairly immune to interfering signals.

Circuitry and construction

The tangible circuit in Figure **3** is not much more complicated than the principle of Figure 2. C1 takes on the role of C_L and at 100 nF, is not really much larger than the capacity of the sensor pads. To keep current under control we have introduced three more resistors, R2 to R4. PB1 lights the LED when the capacitance of the pad rises above the threshold value. Simultaneously T1 is activated via PB0. The open-drain output of the circuit acts as a normally open switch and

Variations and observations

Firmware first: this is written in BASCOM and occupies just over 1/3 of the memory of an ATtiny25. Source and hex code files can be downloaded free from the Elektor web page for this article [1]. The code works as follows: at switch-

About the authors

Martin Jepkens is an electrical engineer and Executive Director of the firm ME-Engineering, which specializes in industrial process automation. His current hobby project is the construction and optimization of a self-balancing scooter. Thomas Scherer has been one of Elektor's contributing authors for more than 30 years. At the moment he is refining

his home-built scooter and is amazed how complex the project has become in each of its aspects.
Daniel Wunsch used to develop household appliances in earlier years and is now employed by a prestigious automobile manufacturer. His homebrew scooter has the best riding characteristics noted yet.

on, the present capacitance of the pad is measured for calibration purposes (it will depend on the specific installation). The threshold value, at which operation of the proximity switch is recognized, is determined in the code as 2/3 of the switch-on value. Because the code listing is commented in great detail, you will find it easy to alter this threshold. Higher thresholds make the circuit more sensitive — although you should not overdo this if you want the circuitry to remain disturbance and error-free.

Needless to say, you can alter the code to suit other microcontroller types or for smaller sizes of pad (e.g. sensor switches). Smaller pads mean reduced capacitance, meaning you will need to reduce the value of C1 proportionally if you don't want to risk excessive numbers of cycles and potentially a variable overflow. You could also connect several pads to a microcontroller. In principle this would require just one pin A for all pads and a separate pin B for each pad.

If you are installing this as a foot switch in a Wheelie, you will need two PCBs, one for each foot. You will also need to check whether each footboard is sufficiently pliant to deflect approx. 1 mm when more than 33 lbs (15 kg) is resting on it. Finally you don't want the gizmo refusing to activate until your full weight is standing on it. Take care too that the proximity switch is calibrated properly when you switch on. At this time neither foot should be on the footboard.

The circuitry can also operate without difficulties using 3.3 V technology. The only alteration is using a 3.3 V version of IC2.

The QTouch technology used is protected by patent but the patentees Atmel have made a QTouch Library available royalty-free (see [2]). Use of the Atmel library entails the use of C, however, which hinders implementation on the very small controllers, which not have so much memory. Patent rights do not affect installation for private purposes but we must warn against commercial use of our circuit!

For Wheelie installations gluing the PCB is preferable to fixing with screws, to avoid any projecting parts causing disruption between footboard and PCB. On the other hand, screwed fixings make good sense elsewhere. There you can save the need to protect the component side with a layer of rubber.

(120505)

COMPONENT LIST

Resistors
(SMD 1206)
R1 = 1.5kΩ
R2,R3,R4 = 120Ω

Capacitors
(SMD 1206, ceramic)
C1 = 100nF 25V
C2,C3 = 1µF 16V
C4 = 4.7µF 10V

Semiconductors
D1,D3 = BAT42W, Schottky, SOD-123
D2 = LED, red, 1206
IC1 = ATtiny25, SOIC-8, programmed, Elektor # 120505-41
IC2 = MCP1703-5, SOT-223
T1 = BSS138, SOT-23

Miscellaneous
K4 = 6-pin SMD pinheader (2x3), 0.1'' pitch
PCB # 120505-1, see [1]
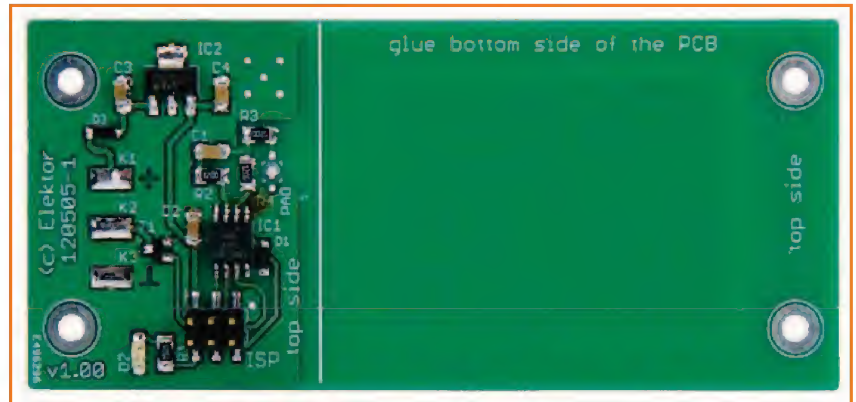
Figure 4.
Prototype of the capacitive proximity switch (component side).

Figure 5.
The rear side of the capacitive proximity switch PCB shows how the sensor surface (the pad) is laid out.
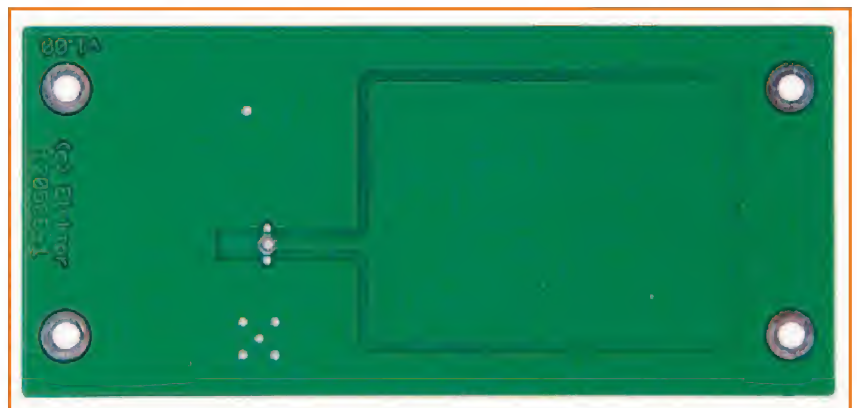
Internet Links

[1] www.elektor-magazine.com/120505

[2] www.atmel.com/products/touchsolutions/
touchsoftware/atmel_qtouch_library.aspx

HangTux

Hangman game on the Elektor Embedded Linux board

By **François-Xavier Maurille, aka FX**
(France) &
Kévin Petit (UK)

Even though everyone (or almost everyone) has heard of Linux, there are still heaps of prejudices to be dispelled in order to increase the penetration of this operating system. Elektor has been doing its bit to help this effort with the Embedded Linux Made Easy series [3] which has been captivating readers for several months now. Here's a concrete little application for the associated Linux board, to show that the little penguin's not as fearsome as you might have thought.

Specifications

- Multiple dictionaries available
- Possibility to choose the level of difficulty
- Number of errors displayed on a bargraph

During his internship in the Elektor lab [1], one of François-Xavier's tasks was to design an application of the Elektor Embedded Linux card. He started with a project dear to his heart for an electronic drum kit, but had to abandon it because he wasn't able to get the performance he was looking for. That was when he had the idea for this game of Hangman.

Every schoolchild knows that Hangman involves getting a player to guess a word, one letter at a time. The player must make as few mistakes as possible in choosing the letters. Each time they suggest a letter that does not appear in the word to be found, one piece of the gibbet or one part of the hanged person is drawn. If the player finds the word before the drawing is finished, they win.

If the drawing is finished before the player finds the word, then they swing for it.

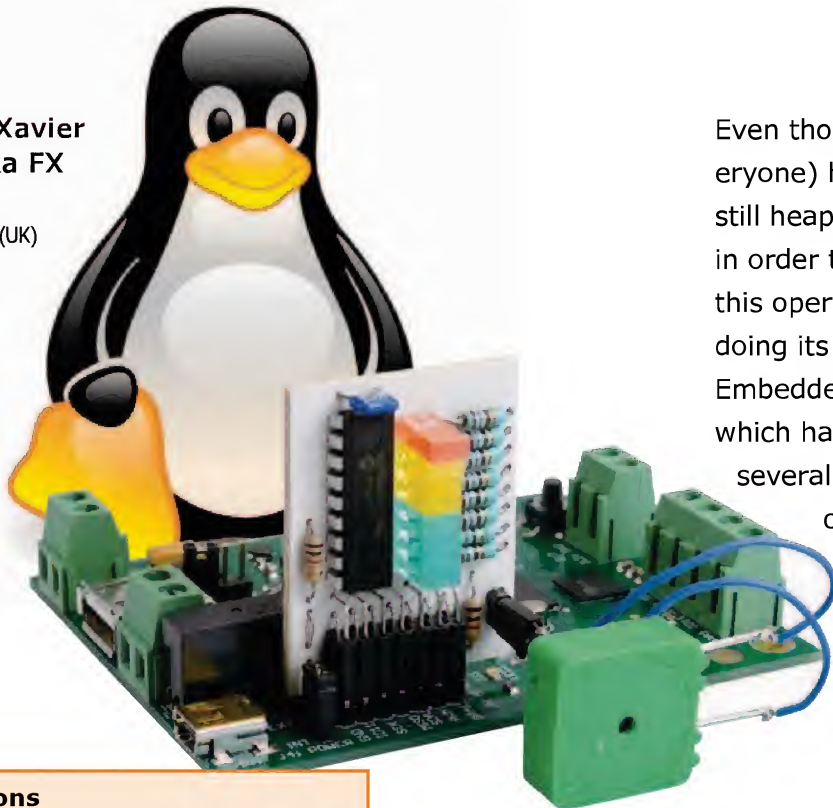Our version has several levels of difficulty, and because of Elektor's presence worldwide, different dictionaries let you choose to play in your own language, or another. The user interface is a Linux terminal and everything has been done to make the game easy and enjoyable to use.

Hardware

In order to improve the user interface and demonstrate the possibilities of the Embedded Linux card, FX decided to develop a small extension (**Figure 1**). To represent the hanged man, the original idea was to use LEDs to draw the outline of the Linux mascot, Tux the Penguin. A cute idea, but ultimately not feasible because of the large number of LEDs and their high current consumption. As it was necessary to allow for

the possibility of powering the card via the USB port, the final choice was a simple bargraph LED (as used for VU meters) which lets us display the number of errors.

As FX felt the number of general purpose input/output lines (GPIO) available on the Linux card interface connector was limited, he opted for a solution that expands these lines using a chip communicating over the SPI bus. So we're going to be using the Linux card's SPI and GPIO modules.

The PWM output of the card has also been made accessible via two pins (PWM and ground) on the extension card, in order, for example, to produce a noise via a sounder, but this feature has not been implemented in the version given here. An excellent opportunity for you to do some experimenting yourself!

The expansion chip chosen is an 8-bit MCP23S08 from Microchip; easy to use, very cheap, and available in a DIP18 package, which makes prototyping easier. The dialog with the MCP23S08 consists in exchanging 24-bit words over the SPI bus. The first byte lets us address the chip and specify if we want to perform a read or write operation. Here, it will always have the value 0100 0000; access is always going to be in write and the two address bits A1 and A0 are both 0 (IC pins 4 and 5 are connected to ground). The next byte corresponds to the register address we want to write to. The only registers our interface board uses are IODIR (I/O Direction), which allows the pins to be configured as outputs, and OLAT (Output Latch), tasked with latching the required logic level on the output pins. The CS (Chip Select) signal is controlled via pin IO11 on the Linux card.

Software

The software that runs this project is written in C. In order to make up for the time lost on the drumkit project he'd originally chosen as an application for the Embedded Linux card, the author has actually recycled an earlier project, but with a number of improvements like the customized word lists, as well as the levels of difficulty.

So that the game can be played in more than one language, FX has chosen to store the words in different dictionaries, one for each language. These take the form of text files with a .dict extension, containing one word per line (a maximum of 100 words per dictionary). To creating your own dic-

tionaries, all you have to do is put your .dict file in the 'dictionary' directory. The def.dict file is used to store the default dictionary.
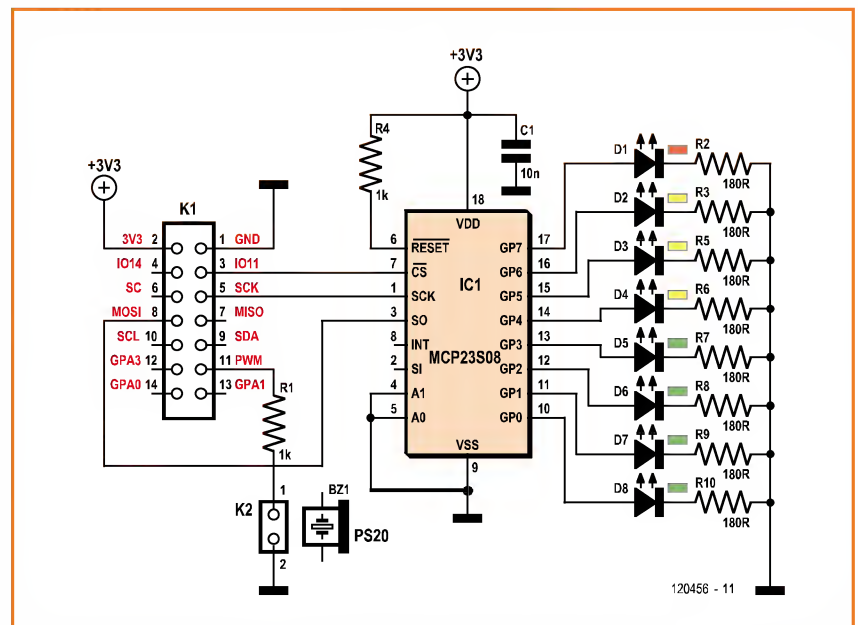
Figure 1.
Circuit for the display card.

There are several solutions for controlling the GPIO lines on the Elektor Embedded Linux card. The first (and most rudimentary) consists in using the command interpreter's 'echo' command to write the desired values to the appropriate file system entry point. Used from within a C program, this is neither elegant nor practical, performs poorly, and above all is not very reliable. We could equally use the 'fprintf' function on the same file, but that's hardly any better.

However, under Linux there is a special file called

COMPONENT LIST

Resistors
R1,R4 = 1kΩ
R2,R3,R5–R10 = 180Ω

Capacitors
C1 = 10nF decoupling

Semiconductors
D1 = LED, 5mm, red, rectangular
D2,D3,D4 = LED 5mm, yellow, rectangular
D5–D8 = LED 5mm, green, rectangular
IC1 = MCP23S08

Miscellaneous
P1 = 2-pin pinheader, 0.1'' pitch
K1 = 14-pin pinheader (2x7), 0.1'' pitch
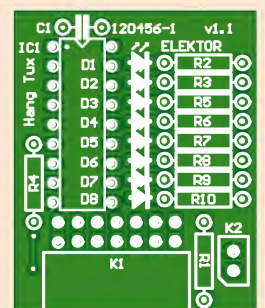
/dev/mem which lets us access the system memory directly, provided, of course, that your program has the appropriate permissions. The usual way this is used from within a C program consists in opening this file and mapping it into memory using the 'mmap' function. In so doing, you obtain a pointer which, if you dereference it, will allow you to access any memory location. By way of an example, here's how to write the value 'val' to the memory location 'offset':

```
*(ptr + offset) = val;
```

Using this technique, which performs better and is more reliable than the previous two, FX accesses the 'IOCONFIG' GPIO configuration register. Setting the various I/Os high or low is achieved in the same way, by accessing the relevant registers. In terms of the SPI, the 'ioctl' system call is used on the special file corresponding to the SPI bus (/dev/spi0), in particular for configuring the number of bits per word. Data is written to the bus via the 'write' system call.

As for the graphics, there's a choice of two shapes available: the traditional matchstick man or, since it's Linux, Tux the Penguin (see the 'graphic.c' file in the source code [2]).

Setting up and use

In terms of hardware, it's very simple, as all you have to do is connect (with the power off, of course!) the extension card to the Elektor Embedded Linux card using the expansion connector.

Before running the software for the first time, you'll need to create the special file /dev/mem that is going to allow it to access the GPIO registers. To do this, all you have to do is type the following commands into a terminal:

```
mknod -m 660 /dev/mem c 1 1
chown root:kmem /dev/mem
```

You will then be able to start the game, still using a terminal. Go first into the folder where you've put the binary that you've compiled or downloaded from our site [2]. You'll then be able to

---

Listing 1. Game algorithm

```
Precondition: - correctly configured extension card

Start
  IF no dictionary has been chosen THEN
    Load default dictionary "dictionary/def.dict"
  End IF
Loop WHILE user wants to play
  Choose a random word from those in the dictionary loaded and hide parts of it according to the level
  Loop WHILE the word has not been found OR the number of errors has not reached 7
    Ask player for a letter
    IF the latter has not been suggested before THEN
      IF the letter is present in the word THEN
        Display the letter in the word
      IF NOT THEN
        Increment the number of errors
      End IF
    End IF
    Refresh display and bargraph according to errors
End loop
  Display message, depending on whether the word has been found (WIN) or not (LOSE)
  Suggest a new game
End loop
End
```

start the game using the command *./hang_tux*.

Once running, the main menu is displayed to let you start a game, change the dictionary or level of difficulty, or quit the program.

When the user chooses to start a game, the dictionary is loaded into a table in memory (if this hasn't already been done) and a word is chosen at random. More or less of the word will be revealed, depending on the level of difficulty. The game can now start. The player suggests a letter by typing it in and then confirming using the 'Enter' key. If the letter has already been chosen before, the message "key already pressed" appears. Otherwise, the software checks if the letter appears in the word. If it is, then the player is allowed to enter another letter. If not, some funny messages will be displayed, the error counter incremented, the stickman or penguin will be (re)drawn, and the bargraph status updated. The game ends either by hanging following seven errors (**Figure 2**) or when the player wins by finding the hidden word. In either case, you can bet your bottom dollar that most players will want to have another go, and another, and another... You'll find the game algorithm in **Listing 1**.

In the main menu, users can choose the dictionary they want to use. The presence of the file X.level indicates that the user has chosen level X. Thanks to this file, the game remembers what level the user has chosen.

Modest and encouraging

We hope that this modest little game will have shown you that it is easy to develop for the Elektor Embedded Linux card and will encourage you to have a go at some projects of your own. Linux is constantly gaining ground in the world of electronics, so don't miss the boat, and watch out for Elektor Projects [4]!

(120456)

Internet Links and References

[1] www.elektor-magazine.com/120609

[2] www.elektor-magazine.com/120456

[3] www.elektor-magazine.com/120026

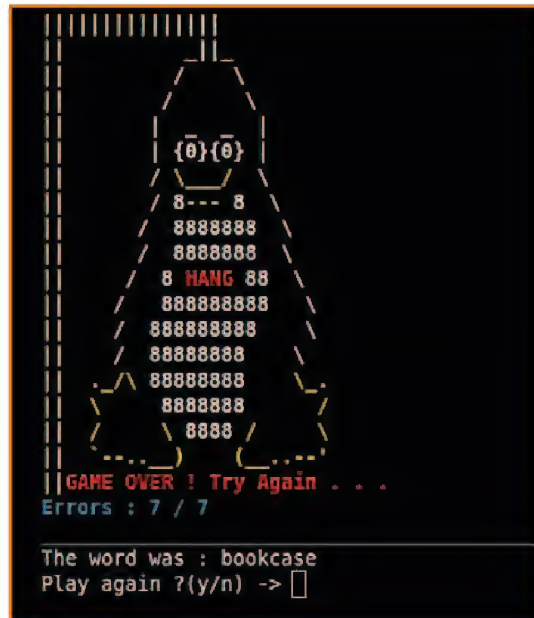Embedded Linux (1) Elektor issue 407 May 2012

[4] www.elektor-projects.com/

Figure 2.
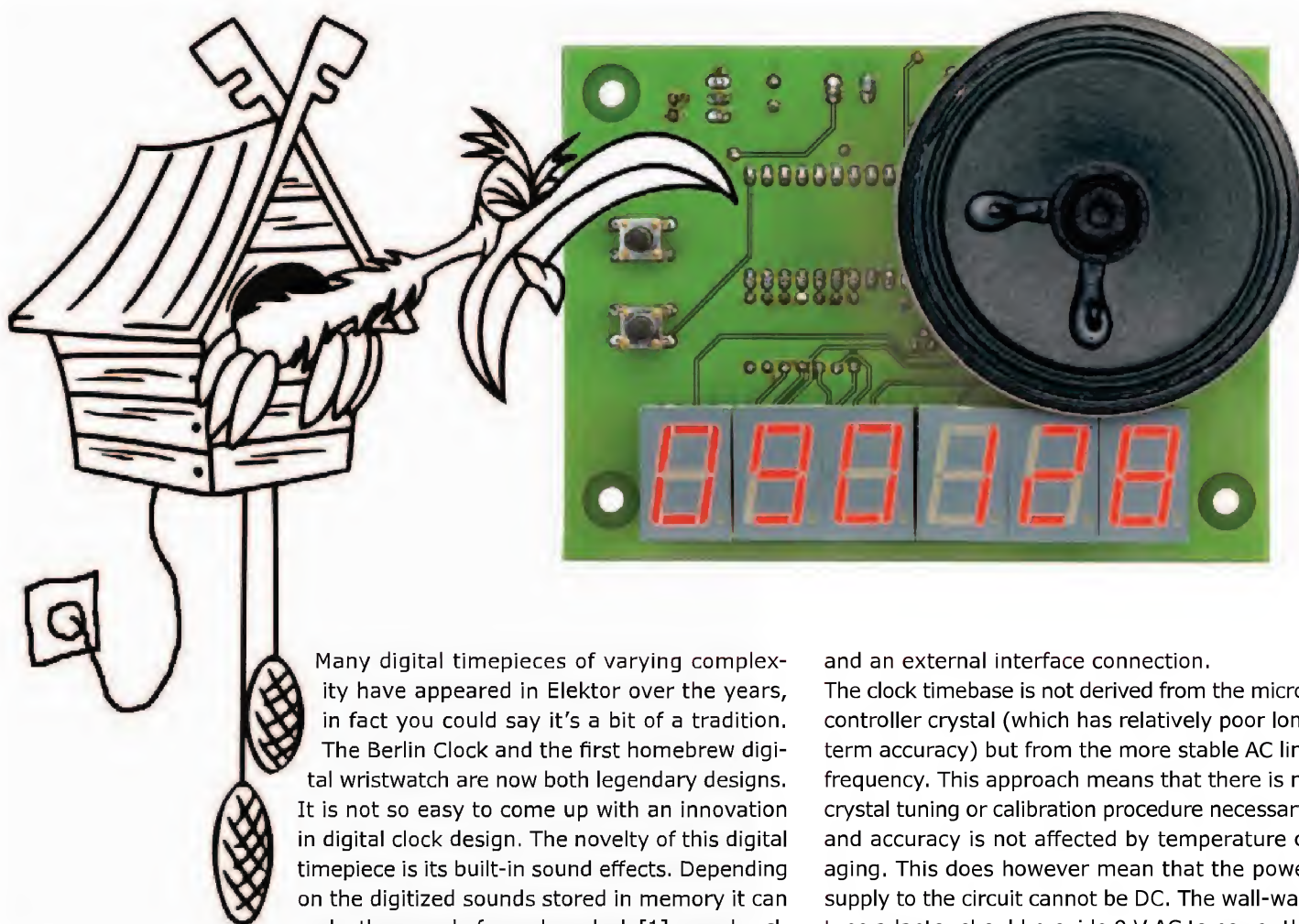Screen snapshot of the terminal after a game has been lost.

A Striking Digital Clock
Cuckoo or chime?

By Martin Ossmann
(Germany)

Surely not another digital clock? Well this one is simple to build and a bit of a novelty. It strikes every quarter hour and hour, and the sound it makes? Well you could go traditional and choose a cuckoo or a chime; in fact it's really up to you…

Many digital timepieces of varying complexity have appeared in Elektor over the years, in fact you could say it's a bit of a tradition. The Berlin Clock and the first homebrew digital wristwatch are now both legendary designs. It is not so easy to come up with an innovation in digital clock design. The novelty of this digital timepiece is its built-in sound effects. Depending on the digitized sounds stored in memory it can make the sound of a cuckoo clock [1] or a church clock. On each quarter hour a chime strikes one, two and three then on the hour another chime sounds indicating the hour.

The circuit

Ticking away at the heart of the circuit in **Figure 1** is an Atmel ATmega32 microcontroller. In addition to the obligatory 16 MHz clock crystal and 5 V voltage regulator IC1 the microcontroller has a PWM signal amplifier, six 7-segment LEDs with transistor drivers, two push buttons (S1 and S2)

and an external interface connection.

The clock timebase is not derived from the microcontroller crystal (which has relatively poor long term accuracy) but from the more stable AC line frequency. This approach means that there is no crystal tuning or calibration procedure necessary, and accuracy is not affected by temperature or aging. This does however mean that the power supply to the circuit cannot be DC. The wall-wart type adapter should provide 9 V AC to power the circuit. In addition to providing power to the regulator the low voltage AC signal is connected via R1 to the microcontroller's interrupt input INT0 (PD2) to give the timebase signal.

The voltage regulator consists of three components and is quite straightforward. D1 provides simple half wave rectification and C1 acts as a reservoir to smooth out the half wave pulses. The fixed voltage 7805 regulator provides 5 V to power the circuit.

Figure 1.
A microcontroller and six
LED displays.
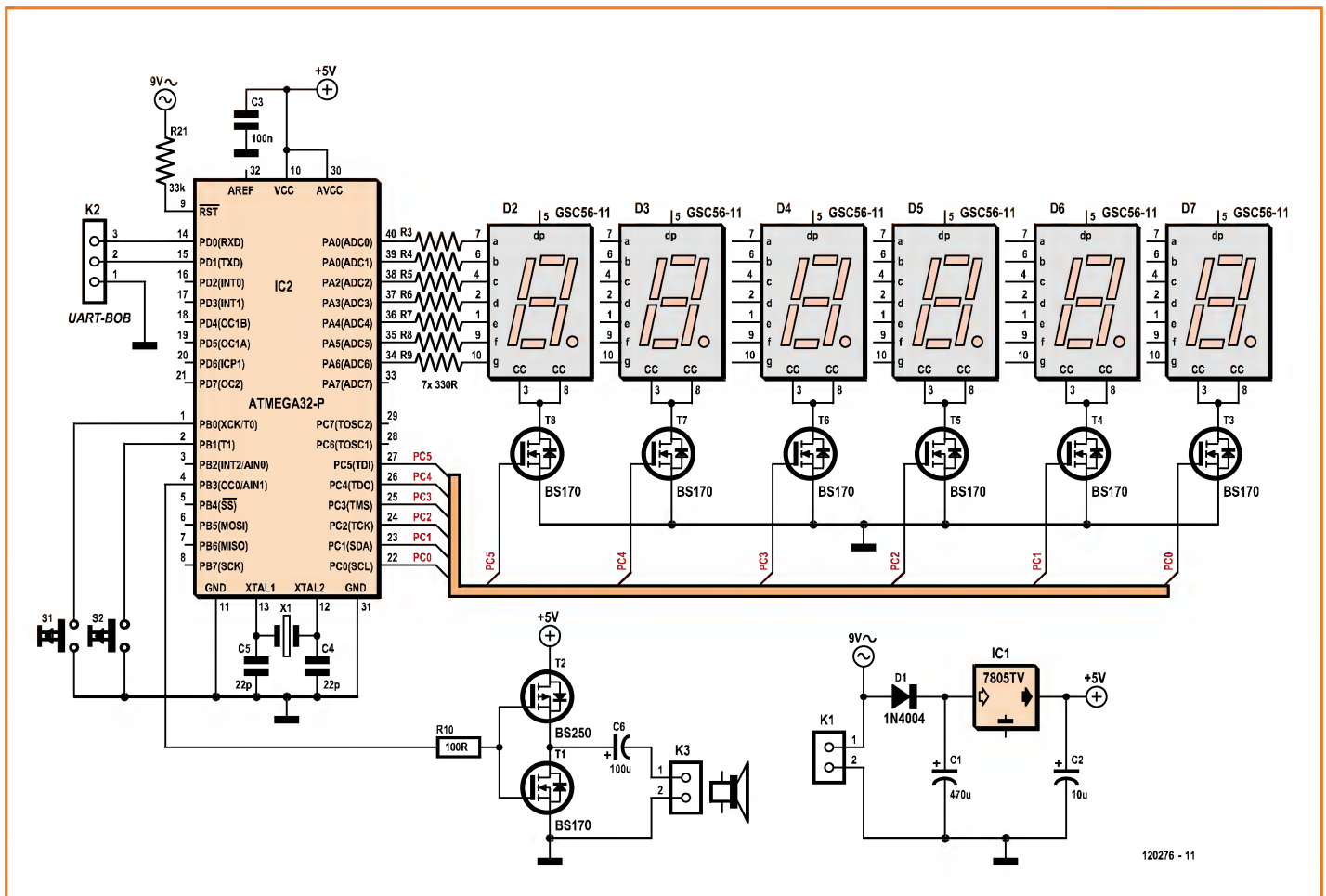
Inputs and outputs

The ATmega output ports cannot drive too much current so an amplifier stage for the loudspeaker is included in the audio path. This consists of just two MOSFETs (T1 and T2), a P-channel type BS250 and an N-channel type BS170. The output audio has a level which is sufficient for most applications.

Time is displayed in the format hh:mm:ss using six 7-segment LEDs. Again we have employed N channel driver transistors type BS170 (T3 through T8) to handle current from the common cathode connections. Current to the segments is restricted by resistors R3 to R9.

Push buttons S1 and S2 allow the displayed time to be set. With the clock operating in its normal mode a press of S2 will start the setting mode. The tens-of-hours number begins flashing. Pressing S1 will now change the number, when the

desired number is displayed press S2 again. The number is accepted and now the hour-units position starts flashing and so on. Continue this for the complete time display until a final press of S2 returns the clock to normal operation.

Wave sounds

For analogue output signals Timer0 is used in fast-PWM mode. The PWM frequency is given by 16 MHz/256 = 62.5 kHz, easily above the ear's upper frequency threshold. The sound is played at a rate of 11025 samples/second, and is controlled by Timer1 which counts to 16 MHz/11025 Hz = 1451. The sound is stored in the program memory in a field called sound as 8-Bit values. As supplied the firmware contains samples of a chime sound. The interrupt routine for Timer1 Sound-Interruptroutine is given below. The sound playback begins by simply setting the value of samplPtr to 0. The second chime is achieved by just increasing the play speed.
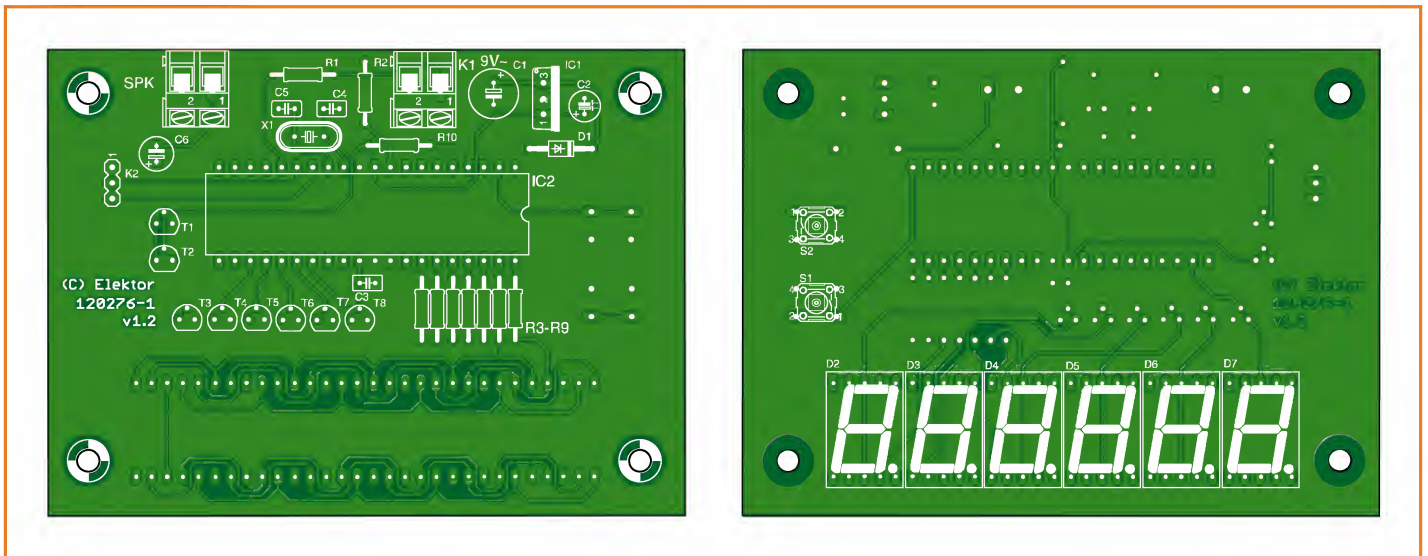
Figure 2.
Component placement on the double-sided PCB. The displays and pushbuttons are mounted on the other side.

It is also possible to use other sounds such as a cuckoo. The sound must be recorded in stereo with a sample rate of 11025 Samples/s and stored as a wave file. Now using the Java program convert1.jar this file can be converted to an Include file for the clock project and used when the firmware is next compiled. The software for this project includes batch files which show how this is done. There is approximately 24 K memory free to store sounds on the ATmega32; this corresponds to 24000/11025 = 2.1 s.

Interfacing
The serial interface at connector K2 provides a connection for TXD (PD1) and RXD (PD0) of the microcontrollers in-built USART. This can be used, for example to connect a USB/TTL converter. The serial signals are at TTL level so direct connection to the serial RS232 port of a computer will not work without RS232 level shifters.

The interface performs two functions namely outputting the time and setting the clock time. Both of these operations are performed with a terminal emulator program (e.g. HyperTerminal) running on the PC using 19200 Baud communication speed. The time is updated every second, To change the displayed time just enter the new time (hours first) and then press the 's' key to set the clock. For example:

Clock 09:01:52 type 's' to set clock to: 00:00
Clock 09:01:53 type 's' to set clock to: 00:00
...

COMPONENT LIST

Resistors
R1 = 10kΩ
R2 = 33kΩ
R3–R9 = 330Ω (or 7-R DIL array)

Capacitors
C1 = 470µF 16V
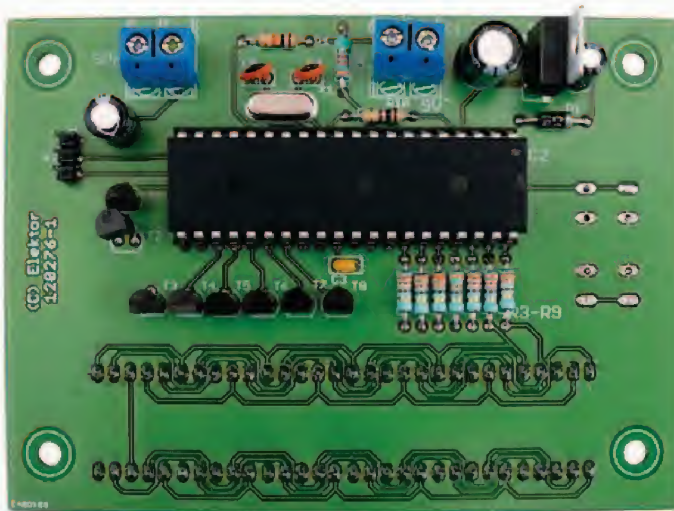C2 = 10µF 16V
C3 = 100nF
C4,C5 = 22pF
C6 = 100µF 10V

Semiconductors
D1 = 1N4004
LD1–LD6 = 7-segment display, Kingbright type SC56-
11SRWA (red)
IC1 = 7805
IC2 = ATmega32P, programmed
T1,T3–T8 = BS170
T2 = BS250

Miscellaneous
X1 = 16MHz quartz crystal
S1,S2 = push button
K1,K3 = 2-way PCB terminal block
K2 = 3-pin pinheader
40-pin IC socket

Enter the desired time as '0845' from the terminal. Now when 's' is pressed:

Clock 09:01:56 type 's' to set clock to: 08:45
Clock 09:01:57 type 's' to set clock to: 08:45
...

A press of 's' updates the clock to the last entered value:

Clock 08:45:00 type 's' to set clock to: 08:45
Clock 08:45:01 type 's' to set clock to: 08:45
...

The ATmega32 does allow in-system programming but the programming interface has not been implemented on the PCB. In order to reprogram the microcontroller (e.g. to store new sounds) it will be necessary simply to remove it from its socket and plug it into a programming adapter.

Assembly

Fitting all the components to this double-sided PCB (available from the Elektor shop [2]) should present no problems at all thanks to the use of standard leaded components (**Figure 2**).

The displays and both push buttons are fitted to the rear side of the PCB.

The microcontroller should be mounted in an IC socket to simplify removal in order to reprogram new sounds. The ATmega32 can be ordered ready programmed from the Elektor shop (go to the Elektor web page for this article), alternatively the C source code can be downloaded for free from Elektor [2] which enables you to program your own controller.

(120276)

Internet Links

[1] www.fuglar.no/galleri/lyder/Cuculus.canorus.mp3

[2] www.elektor-magazine.com/120276

Sound Interrupt routine

```
prog_int8_t sound []={
137, 138, 140, 135, 138, 124,
........
........ 129, 128, 124, 127,
} ;

#define Nsamples 18000

ISR(TIMER1_OVF_vect) {
if ( samplePtr<Nsamples){
OCR0=pgm_read_byte( sound+samplePtr);
samplePtr++ ;
}
```

Caught in the Ring Light

LED Ring Light for close-up camera work

By **Vincent Himpe**
(USA)

This Ring Light is an illumination source used with webcams, microscopes and photo cameras. It is mounted around the lens to provide an even illumination when a subject is very close. Ring lights are commonly used for portrait or macro photography, or to illuminate objects under an inspection microscope. Here we build a powerful ring light ourselves from a kit of parts.

Traditional ring lights use a ring shaped fluorescent tube and are thus restricted to a preset power output unless you want to employ complex and expensive fluorescent dimmers.

The availability of high powered white LEDs allows for a better approach. The ring light presented in this article uses an array of 36 white LEDs controlled by a programmable constant current boost converter specifically designed to drive LEDs.

Anatomy of an LED

The LEDs used in this design are Cree type CLA1B surface mounted devices in PLCC4 package (see also **Figure 1**). With a body size of merely 3.2 by 2.8 millimeters these LEDs can produce up to 13.0 lumen with a dispersion angle of 120 degrees. The maximum continuous forward current is 80 mA, but in this application we run them well below that spec.

The driver circuitry

To drive these LEDs we use a Micrel type MIC3289 LED driver chip. This device is essentially a pulse-width modulation boost convertor that takes a low input voltage and creates a high output voltage (**Figure 2**). The chip measures the current drawn from the output, and adjusts its output voltage so that the preset current flows through the load, in this case the LED chain(s). Where a normal boost converter produces a fixed out-

put voltage, this device essentially produces a variable output voltage to reach a fixed current through the load. Since LEDs are current controlled devices, that's the ideal solution.

The MIC3289 has some other features on board that were specifically designed with LED control in mind. A 16-step current control with a logarithmic scale is built in. The human eye perceives brightness on a logarithmic scale, so this current control largely matches the sensitivity curve of our eyes.

Most of the circuitry is integrated in the chip, and externally only a current sensing resistor and an inductor are required. The chip switches at a very high frequency (1.2 MHz typical) thus allowing for tiny little inductors. A digital control pin allows the user to control the brightness and on/off state of the boost converter.

The boost convertor can produce a high voltage, possibly destroying the LED chain if something went wrong. Therefore, a built in voltage detector will shut down the booster before any harm can be done.

Two versions of the driver are available: 16 or 24 V output voltage. As the forward voltage drop across a single white LED can be up to 3.8 volts and we have six of them in series, we need the 24-volts version. A quick peek in the datasheet tells us we need the MIS3289-24YD6 part.

The MIC3289 senses the current as a voltage drop across a resistor linked to the FB (feedback) input pin. Maximum output is achieved when this input reaches 250 millivolts. The 1-ohm sense resistor produces 250 mV when 250 mA flows through it. Spread across the six LED chains, that's roughly
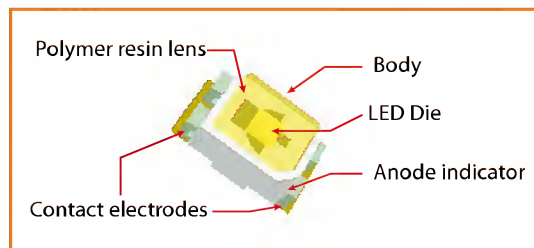

Figure 1
The anatomy of an LED.
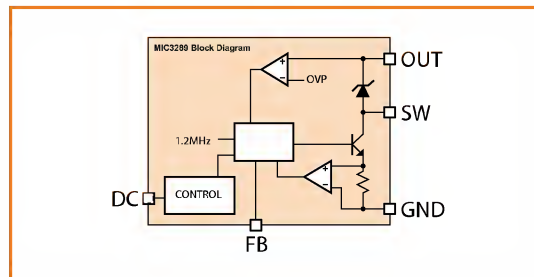
Figure 2.
Internals of
the LED driver IC.

42 mA per chain, well below the 80 mA maximum specified for the LEDs. The MIC3289 is capable of putting out well over 500 mA, so you can play with the maximum brightness by lowering the 1-ohm shunt to as low as 0.5 ohms. You do need to take care not to drive the MIC3289 in thermal shutdown as it will heat up. The chip has internal current sensing circuitry to protect its switching transistor as well, so you can't really destroy it.

Microcontroller
The circuit printed in **Figure 3** is reproduced from the *Mastering Surface Mount Technology* book published by Elektor. U1 takes care of the *heavy lifting* of the power in conjunction with inductor L1 and current sense resistor R8. The capacitor

Figure 3.
The core of the circuit consists of U1, the components around it are 'just' controlling (U2, SW1-4), decoupling (C1-11), acting (D1-36) and supporting (R1-5).

Projects

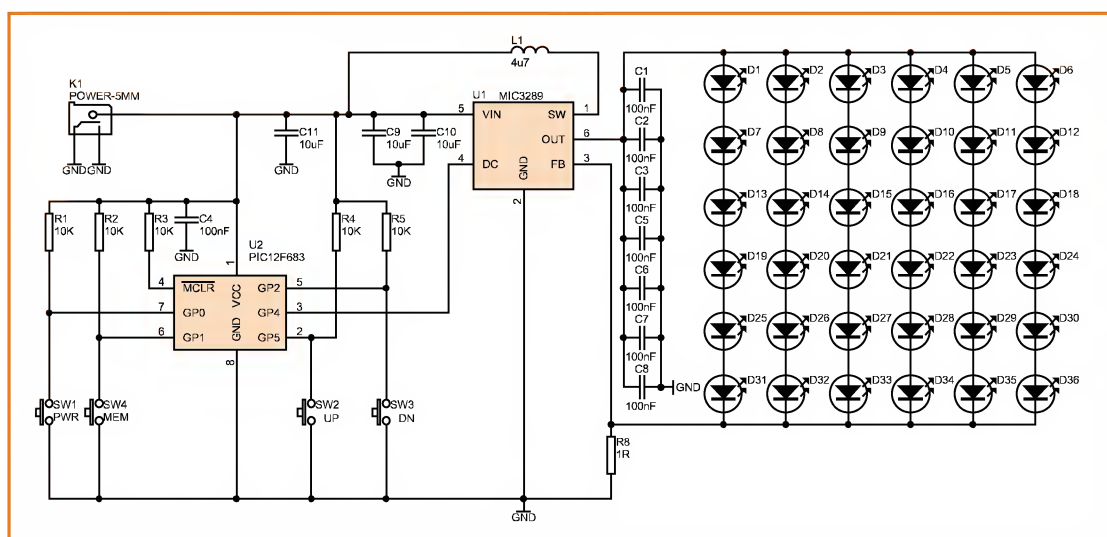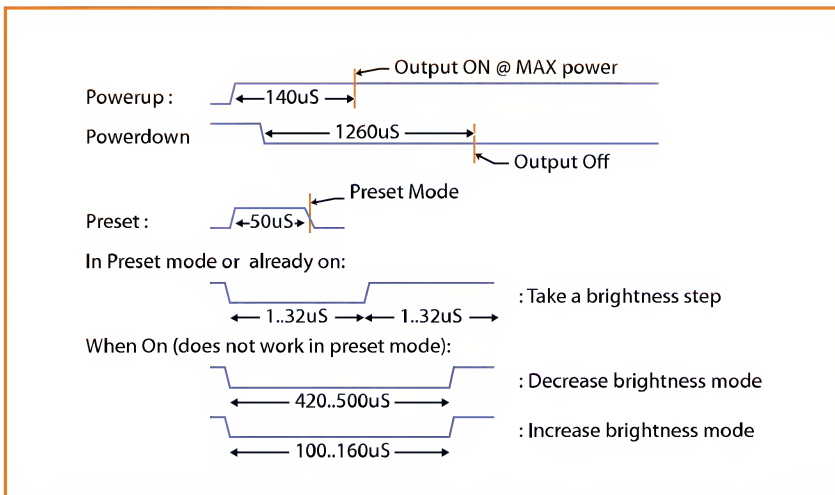bank formed by C1, C2, C3 and C5-C8 provides adequate smoothing of the output voltage of the MIC3289. The six LED chains each consist of six LEDs and are simply wired in parallel.

Microprocessor U2 is used to generate the control bit stream for the MIC3289. Four switches together with their appropriate pull-up resistors are attached to its digital inputs. Key debouncing is implemented in software. The switches allow for adjusting, switching on and off, and storing

---

Bonus project: LED Tester

Included with the LED Ring Light kit is a small practical gadget for checking the polarity of SMD LEDs. This kit is a nice little project to practice your SMD assembling skills. Make sure you practice applying the solder paste using the solder stencil and align components correctly before soldering them onto the PCB. A list of parts is available at [1].
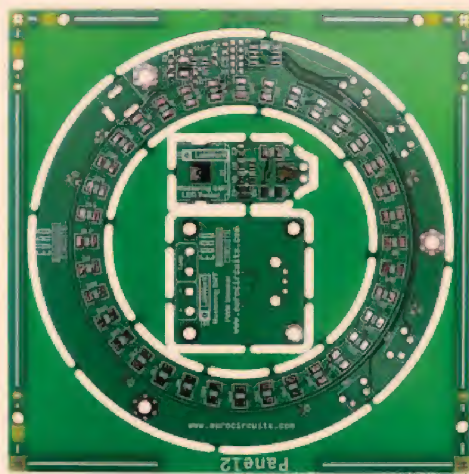
Bonus Project: PWM Controller for Light Bulb

The second bonus project comprises a PWM controlled dimmer for resistive or inductive loads such as light bulbs and motors and builds on the skills gained from the LED Tester project. Its input voltage ranges from 5 to 24 V, and it has a soft-start function. A potentiometer determines the duty cycle. A list of parts is available at [1].

---

a preset. Every time the UP or DOWN button is pressed, the light intensity is adjusted accordingly. Actuating the STORE button saves the current brightness setting to EEPROM.

The whole circuit can be powered from a standard 5 V / 1 A power brick via connector K1.

Instruction timing

The communication protocol is unique as it uses just a single wire. Whenever the MIC3289 sees a level change on the DC pin it starts an internal timing system. When the DC pin is switched from Low to High, a timer is started that waits for 140 microseconds. If no further changes occur, the MIC3289 engages its regulator and delivers full power to the LEDs as set by the current sensing resistor.

If the signal on the DC pin changes within the first 50 microseconds of the 140 microsecond window, the MIC3289 enters programming mode. Pulling the DC pin Low for more than 1260 µs turns off the output. This allows the MIC3289 to be used as a simple Full-on/off switch. Simply pull the DC input High and leave it there. After 140 µs the converter will start. Pulling the line Low will turn off the converter after 1260 µs.

When a pulsed signal is sent to the DC input within the first 50 µs after pulling it High (assuming the output was previously off), advanced control commands can be sent to the controller. When the programming mode is entered, another internal counter starts. If the MIC3289 sees a Low signal for 100-160 µs, that is considered the command for increasing the brightness. If it detects a Low signal for 420-500 µs, a request is seen to decrease brightness. Any pulse between 1 and 32 µs indicates one adjustment step has to be taken in the programmed direction. The programmed direction and brightness step are retained for as long as the output is active.

If you power down the MIC3289, then the brightness is reset to Max. and the count mode is set to 'decrease brightness'. This allows for programming a 'preset' during power up. Simply pull the DC pin High and within the first 50 µs start pulsing the DC pin with logic zeros for a duration of 32 µs max. with the number of brightness steps you want to decrease. Leave the pin High and the MIC3289 will start powering the LEDs another 140 µs after the final pulse.

In other words (see also **Figure 4**):

To turn the output on:

Up to 460 lumens
with a dispersion angle of 120 degrees

- Pull the DC pin High for longer than 140 µs.
- When the output is already on:
- Pulling the DC pin Low in-between 100 and 160 µs changes the adjustment mode to INCREASE.
- Pulling the DC pin Low in-between 420 and 500 µs changes the adjustment mode to DECREASE.
- Pulling the DC pin Low in-between 1 and 32 µs adjusts the brightness with one step.
- Pulling the DC pin Low longer than 1260 µs turns the output off.
- To preset at power up:
- Within the first 50 µs after pulling DC High apply the number of steps required (pulses between 1 and 32 µs with a wait time of 1 to 32 µs between them). The device powers up in DECREASE mode and at max. setting.

Examples
Example 1.
At power on we want half brightness. We pull DC High for 40 µs and subsequently send eight Low pulses of 16 µs with a delay of 16 µs after which we leave the DC pin High. 140 µs later the driver start to power the LEDs at half brightness. We used 16 µs as it is half of the 1 to 32 µs
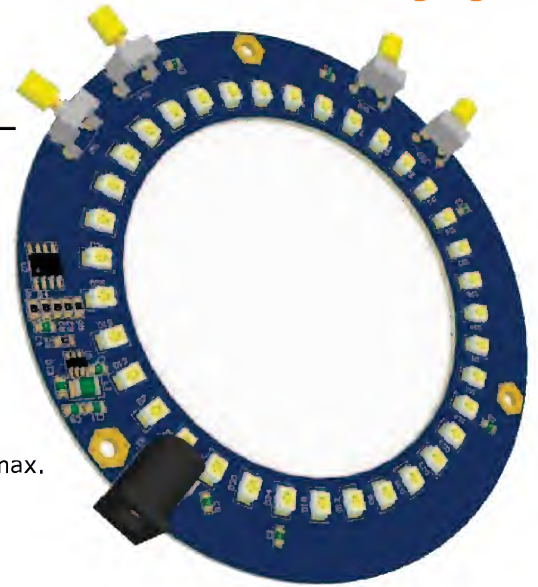
requirement. Since the default mode is power up at max. power, the eight pulses in DECREASE mode reduce brightness 8 steps from max., half of the 16 steps available.

Example 2.
While running, we want to increase the brightness by one step. This is done by sending a pulse between 100 and 160 µs in length to switch to INCREASE mode. Then, a pulse of 16µs is sent to adjust the brightness by one step.

Example 3.
While running, we want to decrease the brightness by one step. Feed a pulse between 420 and 500 µs to switch to DECREASE mode, followed by a 16 µs pulse to adjust by one step.

Microprocessor
We selected a small 8-pin PIC12 micro with onboard EEPROM to control the MIC3289. The code monitors four pushbuttons, and determines the commands to be sent to the MIC3289. The UP and Down buttons generate the appropriate commands to adjust brightness in the desired direction. The microprocessor retains the mode (Decrease/Increase) the controller is in, and

where it is on the step scale, so it is not necessary to send mode changes every time (but it is prudent to do so).

The controller's internal step counters 'roll over', meaning that, when max. is reached it goes back to min. and vice versa. The controlling device (the microprocessor in our case) can block this by keeping track of the current position on the scale, and refrain from sending a command resulting in roll-over when either end position is reached. This is implemented in the firmware produced for the project. The PIC keeps track of the position on the brightness scale and prevents roll-over. The MEM button stores the current brightness setting in the processors' internal EEPROM. The PWR button toggles the light on and off. When the circuit is switched on, the brightness level stored in EEPROM is applied to the light.

PCB design
A comprehensive kit of parts is available for this project (see **inset**) with a few bonuses thrown in. The board is ring shaped so it can be mounted around the lens of the instrument used (camera, microscope). All electronics is installed at the bottom side of the board with the possible exception of the pushbuttons. Three mounting holes allow you to clamp the system onto the lens. You can string rubber bands, use tie wraps or make a custom spring jig to clamp these down. The same holes can be used to mount a diffuser made from a piece of frosted Plexiglas if you so desire. Assembling the board is easy. Start with the passive parts such as resistors and capacitors, followed by the LEDs. Then, install the ICs and finish with the thru-hole pushbuttons and power connector.

Now let your light shine and show us some interesting and well lit close ups of your circuits!

(120701)

Mastering Surface Mount Technology book

If you'd like to read more about this project or the soldering of SMT devices we recommend you order a copy of our book *Mastering Surface Mount Technology*, the second book from Elektor's popular *LabWorX* series. This book takes you on a crash course in techniques, tips and know-how to successfully introduce surface mount technology in your workflow.
Visit **www.elektor.com/labworx** for more information.

Kit available!

A kit of parts to build the Ring Light project is offered via our business partner Eurocircuits. Visit **www.elektor.com/ringlight** to order your kit of the LED Ring Light and get two bonus mini projects (a PWM Dimmer and a LED tester) for free. The kit comes complete with high quality circuit boards, all components including the pre-programmed microcontroller, and, uniquely for Elektor, a complementary solder stencil to take the fuss out of applying SMD solder paste.

Internet Links

[1] www.elektor-magazine.com/120701

[2] www.elektor.com/labworx

Inrush Current Limiter

When you first switch the power to certain AC loads (power transformers or motors for example) it can happen that the current inrush is so large that it takes out the equipment fuse or trips the motor protection circuit. The only way to get the equipment working again now is to replace the fuse or reset the trip device. An inrush-limiter circuit gets around the problem by momentarily switching a high power resistor in series with the load to limit the maximum inrush current. Once the surge subsides this series resistor is bypassed so that the load receives full AC line voltage.

By **Wilfried Wätzig**
(Germany)

The circuit shown here is a current-controlled inrush limiter placed in series between the AC line and the switched load. It uses a coil and a Hall-effect switch to monitor load current passing through a coil wound on a ferrite toroid. The toroid used here is a type FT 114-74 which has a 1.14″ (≈ 30 mm) outside and 0.74″ (≈ 20 mm) inside diameter and a height of 0.24″ (≈ 6 mm). Now for the tricky part: it is necessary to make an air gap by slicing a 1.5 mm slot through the toroid. The ferrite material is very hard so we have shown how this has been achieved in a separate article dot Labs section in this magazine. The slot should be just wide enough to accept the TLE4935L Hall-effect sensor (see photo). A 30-turns winding of 16 AWG (≈ 1.2 mm) enameled copper wire is wound onto the toroid. Load current flowing through the coil produces a magnetic field in the air gap which passes through the Hall-effect switch. This sensor is the digital variant and includes a Schmitt trigger and an open-collector output. When the magnetic

field exceeds the sensor's threshold it switches and pulls the output down to ground. Using the components and coil in this example the threshold occurs at approximately 1 A load current. The active area of the Hall-effect sensor is in the region of 1 mm² which is quite a bit smaller than the 30 mm² cross sectional area of the ferrite ring. Using the parts suggested here it is clear that the sensor will only be measuring a fraction of the magnetic field. A ferrite ring with a smaller cross section would in fact be a more suitable choice.

The circuit is quite straightforward: the inrush limiter is connected to the AC power line at connector K1 while the inrush protected load and series on/off switch is connected to K2. Transformer TR1 together with bridge rectifier B2 and the voltage regulator IC2 provides a stable 12 V supply for the circuit.
The coil L1 and Hall-effect sensor IC1 measures the current flowing to the load connected at the

100759 - 11
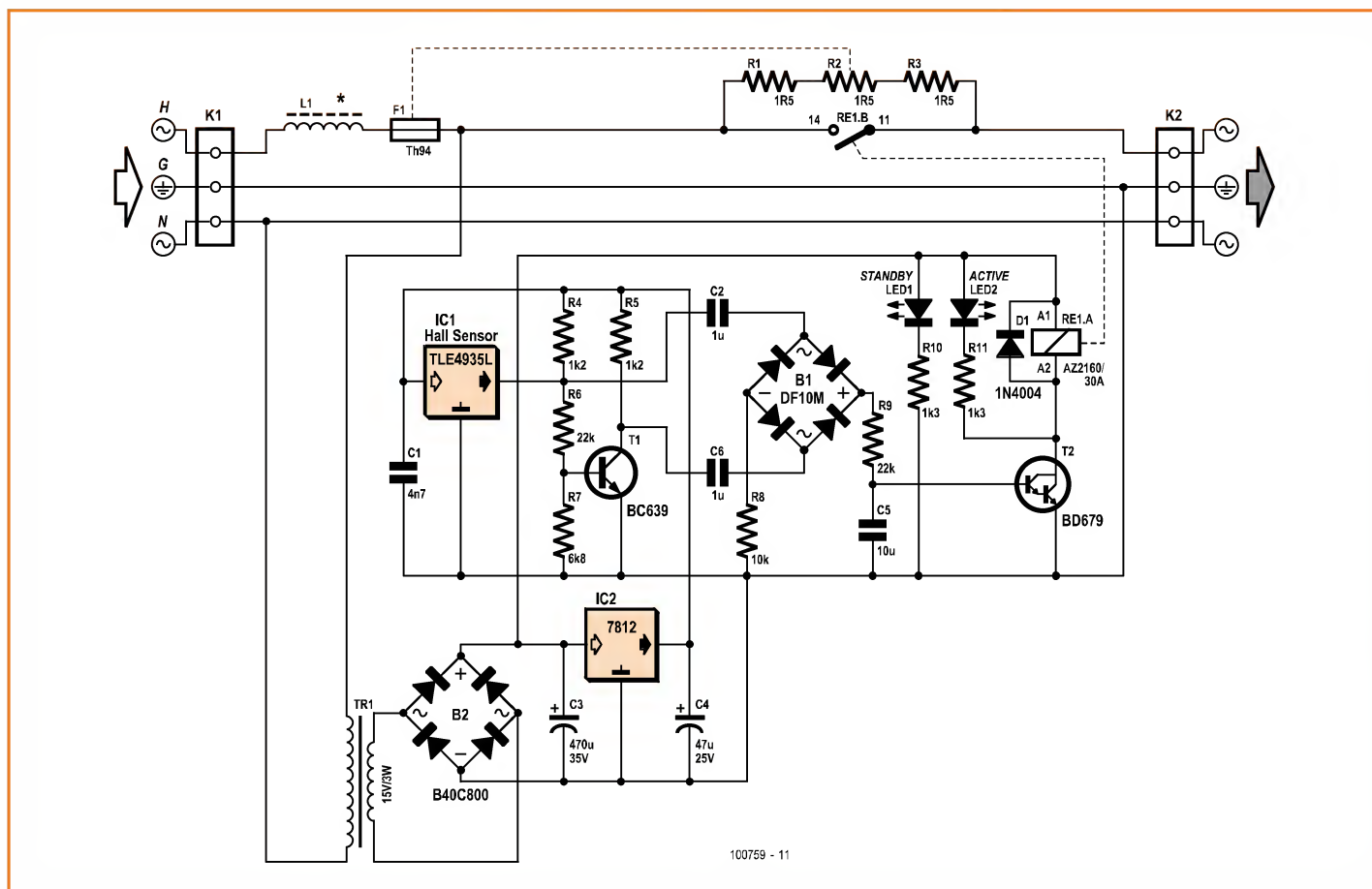
circuit output. When the relay contacts are open the power resistors R1, R2 and R3 reduce the peak load current.

As long as the current in the coil is below the threshold the output of the Hall-effect switch will be pulled high (in the region of 12 V) by the resistor network consisting of R4, R6 and R7. The voltage at the base of T1 will be high enough so that T1 is on. When the load connected to K2 is switched on (and a high current pulse passes through the coil) the Hall-effect switch output goes low and turns off T1. The alternating load current produces a square wave output from the Hall-effect sensor with the same frequency as the AC power line. T1 provides inversion of the signal to drive the full wave rectifier B1. The resultant DC signal charges C5 via R9. The time constant provided by this network produces a delay of approximately 60 ms before the voltage on C5 rises to a level sufficient to turn on the Darlington transistor T2 which in turn energizes the power

relay RE1. During this period the power resistors (R1, R2 and R3) have effectively reduced the peak inrush current.

The surge protection resistor is made up of three series 1.5 Ω power resistors. A surge peaking at 20 A for example, briefly dissipates 600 W in each resistor. The thermal fuse F1 must be in good thermal contact with one of the three power resistors. This failsafe measure turns power off to the complete circuit if the relay contacts fail to close. It is worth noting here that connections to the thermal fuse leads should only ever be crimped or clamped, never soldered.

Lastly and most importantly; parts of the circuit are directly connected to the AC power line. It is vital that all the necessary safety precautions are observed in the construction and safe enclosure of the finished circuit to ensure that accidental contact with any part of the circuit is impossible.

(100759)

Rechargeable Battery Checker

Measure capacity by controlled discharge

By **Martin Ossmann** (Germany)

The author's rechargeable battery collection comprises a large number of cells of various types, ages and states. Surely he cannot be the only person who would find a 'capability meter' for batteries of great value and interest!

A simple way of establishing the capacity of a storage battery is to recharge it and then discharge it again in a controlled manner. With a meter of the kind described you can check accurately whether a battery's capacity can be refreshed after a couple of charge/discharge cycles or it has reached the end of its useful life irreparably. The heart of this circuit is MOSFET T1 and associated shunt resistors R6 to R9. The transistor is driven by a PWM signal in such a way that a predetermined discharge current from the battery (at K2) flows through the shunt. The discharge current and the voltage of the rechargeable battery are both measured, identifying the charge Q drawn (by integrating the current) and displayed in mAh after conversion. From 1.1 V down to the final voltage of 1.0 V the current is reduced linearly from its maximum value down to zero. Time, voltage, current and charge status parameters are displayed continuously on the LCD and also exported over the serial interface, enabling the discharge process to be logged on a PC.

The circuit

If that sounds simple enough, it turns out equally straightforward in reality (that is, the circuit in **Figure 1**). The ATmega88 controller IC1 handles all measurement, calculation and output tasks. The processor is clocked at 11.0592 MHz. An LM7805 supplies the operating voltage of 5 V; the supply voltage at K4 comes from a 9 to 12 V plug-in power supply (wall wart). The 2.5 V reference voltage for the analog to digital converter (ADC) is produced by an LM385-2V5 (IC2). The squarewave voltage at pin 12 (PD6/PWM C0A) produces a negative voltage for adjusting the contrast of the LCD, adjustable with trimpot P1. Jumper JP1 gives you the option of background lighting for the display.

The PWM signal at pin PB1 is transformed using C2 and R3 into a DC control voltage for MOSFET T1. The discharge current is measured using shunt resistors R6 to R9 and regulated with T1 so that the discharge current flows as programmed.
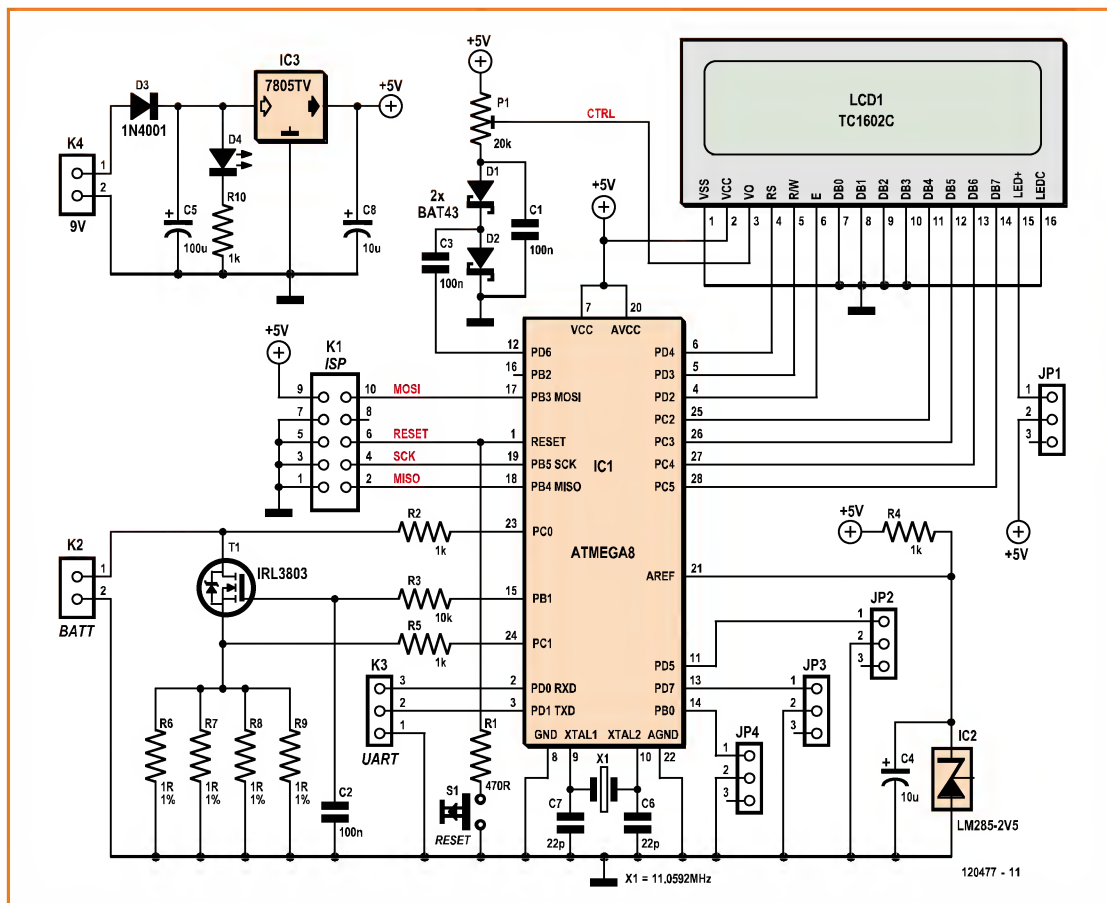
Figure 1.
The battery checker with the discharge circuitry connected to T1.

The two analog channels ADC0 (PC0) and ADC1 (PC1) capture voltage and current. Jumpers JP2 to JP4 let you set the discharge current between 50 mA and 1 A (**Table 1**).

PD0 and PD1 on the controller represent the serial interface. Here for example you can connect a USB-to-TTL converter and export at 19200 baud all parameters in the format

```
T=00043 sek U=1.396 V I=0.502 A
Iset=0.500 A Charge=0003 mAh
```
(sek = seconds).

Connector K1 on the PCB is an ISP programming interface, enabling you to load new firmware into the controller if required.

Construction and use
For this project Elektor provides a bare PCB and a ready-programmed controller. Naturally you also have the option of doing everything yourself and

as normal you can download the PCB layout and all the software from the website [1].

There is really not a lot to be said as far as construction of the Battery Checker on the single-sided PCB shown in **Figure 2** goes. We advise fixing the seven wire links first; one link (passing below the crystal) needs to be attached to the trace (tracked) side of the PCB. All components are of the 'through hole' type, avoiding the

Table 1. Jumper settings for setting the current			
Current (mA)	JP-4	JP-3	JP-2
1000	set	set	set
500	set	set	not set
250	set	not set	set
100	set	not set	not set
50	not set	set	set

COMPONENT LIST

Resistors
R1 = 470Ω
R2,R4,R5,R10 = 1kΩ
R3 = 10kΩ
R6–R9 = 1Ω 1% 0.5W
P1 = 20kΩ trimpot, horizontal

Capacitors
C1,C2,C3 = 100nF
C4,C6 = 10µF 10V, radial, 2mm/5mm
  pitch
C5 = 100µF 10V, radial, 2.5mm/6 mm
  pitch
C7,C8 = 22pF

Semiconductors
D1,D2 = BAT43
D3 = 1N4001

D4 = LED, red, 3mm
T1 = IRL3803
LCD1 = TC1602C
IC1 = ATmega88-20, programmed, Elek-
  tor # 120447-41
IC2 = LM285-2V5
IC3 = 7805

Miscellaneous
LCD1 = LC display 2x16, e.g. Elektor #
  120061-71
K1 = 10-pin (2x5) pinheader, 0.1'' pitch
K2,K4 = PCB terminal block, 0.2'' pitch
K3 = 3-way receptacle, 0.1'' pitch
JP1–JP4 = 3-pin pinheader, 0.1'' pitch
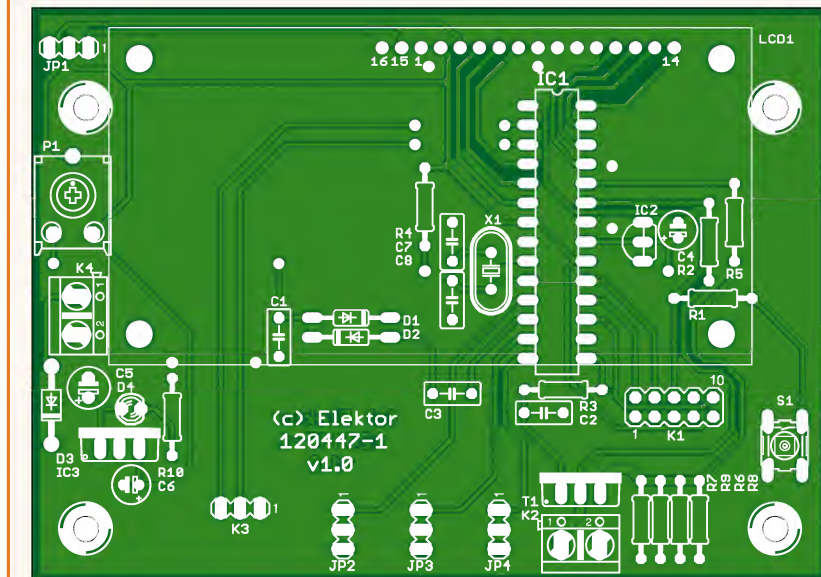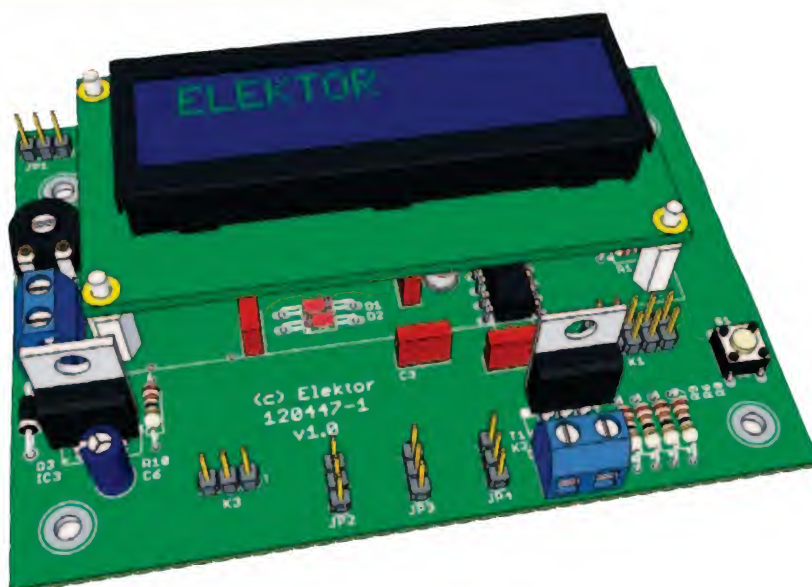S1 = pushbutton
PCB # 120447-1

Figure 2.
The PCB of the battery checker.

Figure 3.
Constructed PCB viewed in SketchUp (you can download the 3-D file created with EagleUp [2] from the Elektor website).

need for SMDs. The display is piggy-backed on the PCB but in a way that still leaves the control interface (connections, press-buttons and jumpers) accessible.

Thanks to the LM385 reference, no alignment is necessary. The fuses in the controller are set for it to run using the external crystal. Transistor T1 requires a small heatsink to cope with the maximum rating of 1 A × 1.3 V = 1.3 W. Shunt resistors R6 to R9 should be 1 percent types to ensure that current measurement is reliably accurate.

After attaching a freshly charged 1.2 V cell to K2 (connect a battery holder here), the reset button S1 is operated to start the timer anew and reset the charging integration. The rest of the process works by itself, with the circuitry adjusting the current to the predetermined value. During the first hour the voltage falls from 1.4 V to 1.2 V. Then for a relatively long time the voltage remains at a constant level of 1.2 V, as you would expect. Finally the voltage drops once more. When it reaches 1.1 V, discharging begins to reduce the discharge current. At the same time the discharging slows down. At 1.0 V the discharge current is reduced to zero, to avoid over-discharging the cell. When the discharge period is finished you can read the capacity of the battery on the LCD at lower left.

(120447)

[1] www.elektor-magazine.com/120447
[2] http://eagleup.wordpress.com

Realistic LED Candle
With adjustable flame color and wind sensor

To build this cozy, realistic lighting effect
we used just a tiny microcontroller and a bi-color LED.
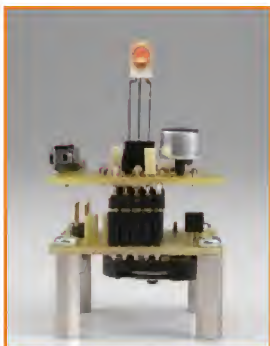
By Jörg Trautmann (Germany)

There are now many different types of LED candle simulators available. The simplest designs use a melody generator chip normally found in 'musical' greetings cards. When the melody plays, the LED (in place of a speaker) flickers apparently randomly. More expensive designs use a wind-noise detector which allows them to be 'blown out'. Most of the designs don't come close to simulating true candle flame coloration: the flame is either a cold yellow or an unrealistic orange and those designs based on greetings card chips produce a flicker which is not very flame-like. Before we begin to design a more realistic candle

with a wind-noise detector and variable flame colour we need to check what a real flame looks like. The light from a normal candle flame burns with a wavelength of 600 nm. The light from a yellow LED is in the range of 565 to 590 nm whilst a red LED is in the range 625 to 740 nm. To achieve the effect of a light source between these two color bands (in the region of 590 nm to 625 nm) it is necessary to adjust the brightness of a red and yellow LED. Sounds like a job for a red and yellow bi-color LED.

A tiny controller
The design is based on an ATtiny13 [1]. This tiny 8-pin microcontroller has two PWM outputs (OC0A and OC0B), which are used here to modulate the

Figure 1.
The two sides of the circuit.
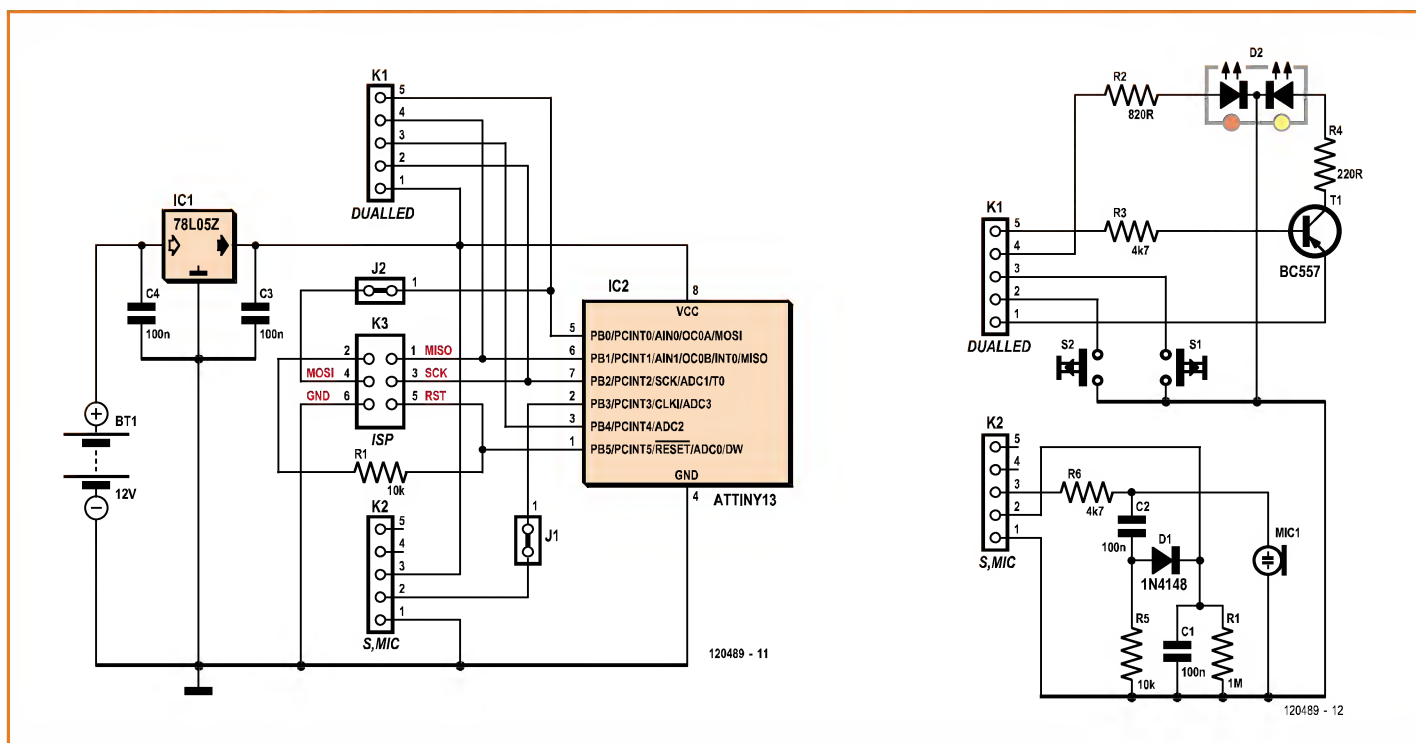
brightness of the bi-color LED. The yellow LED is controlled by PNP transistor T1 while the red LED is driven directly from the controller's port pin. This arrangement acknowledges that for a realistic candle flame the light contribution from the red LED is smaller than from the yellow one. Using a series resistor R2 of 820 Ω limits the current to 5 mA which can be safely sourced from the controller's port pin. The yellow LED has a series resistor R4 of 220 Ω to give a current of 15 mA for greater light output. Transistor T1 is required here to drive the extra current.

Two push buttons are provided to enable the flame color to be adjusted. With S1 held down the amount of red light increases while the yellow decreases. Pressing S2 has the opposite effect. The mark/space ratio is adjustable in 256 steps corresponding to an 8-bit value.

To produce the realistic flame effect a random number generator is used to modulate (by a maximum of 35 %) the mark-space ratio of the PWM signal driving the yellow LED and (by a maximum of 10 %) the red LED. This has been found to give a very life-like soft flicker. Once the optimal flame color has been chosen it is stored in the microcontroller's EEPROM memory so that it will be used the next time the candle is turned on. A small electret microphone capsule Mic1 is used to register wind noise. Diode D1 and capacitor C1 provide some filtering to the signal before it is sampled by the A/D converter. R1 provides a discharge path. The trigger level has been chosen so that a light breeze over the microphone has no effect on the flame. A louder wind noise will cause the LED to flicker and dim. When the noise persists for more than two seconds the 'flame' is extinguished. Now a short puff on the microphone brings the LED flame back to life. The constant `Trigger_value` in the firmware can be changed to compensate for the sensitivity of the microphone capsule. A higher value reduces the candle's sensitivity to wind noise. An ISP programming interface K3 has been included in the design to allow this change to be made to the firmware (and to facilitate any other modifications).

Power to the circuit is provided by two CR2032 button cells. The maximum supply rail voltage for the ATtiny13 is 5.5 volts which is supplied by a low-drop regulator type 78L05. The quiescent current is approximately 1 mA so the batteries should be removed if the candle is not going to be used for some time.

Construction on two PCBs

To make the complete design as compact as possible the circuit has been designed using two small boards which are plugged together. The controller board has two wire links which need to be made while the LED board does not have any. Apart from this there should not be any difficulties fitting the components in place.

A ready-programmed controller is available to order from the Elektor shop. If you are confident enough to program it yourself and posses the necessary tools the software files can be downloaded for free from the Elektor website [2].

(120489)
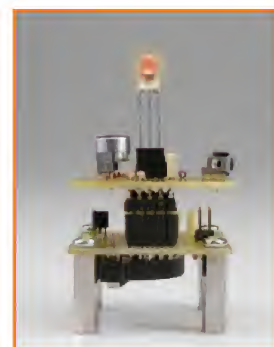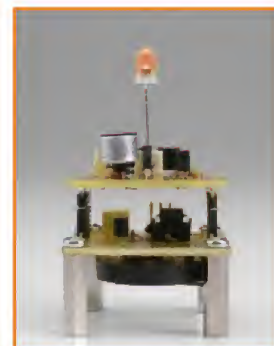
Internet Links

[1] www.atmel.com/Images/doc2535.pdf

[2] www.elektor-magazine.com/120489

COMPONENT LIST

Controller Board

Resistor
R1 = 10kΩ

Capacitors
C3 = 100nF
C4 = 100µF

Semiconductors
IC1 = 78L05
IC2 = Atmel ATtiny13-20PU, programmed, Elektor # 120489-41

Miscellaneous
K1,K2 = 5-pin pinheader, 0.1'' pitch
K3 = 6-pin (2x3) pinheader, 0.1'' pitch
PCB # 120489-1

LED Board

Resistors
R1 = 1MΩ
R2 = 820Ω
R3,R6 = 4.7kΩ
R4 = 220Ω
R5 = 10kΩ

Capacitors
C1,C2 = 100nF

Semiconductors
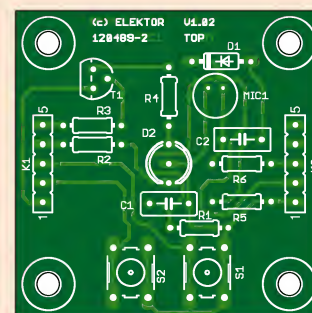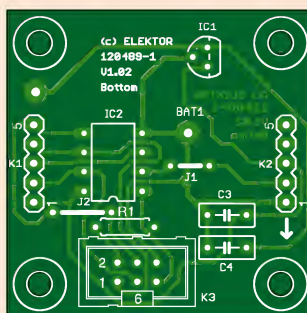D1 = 1N4148
LD1 = bi-color LED, yellow/red
T1 = BC557

Miscellaneous
Mic1 = electret microphone capsule
S1,S2 = push button
K1,K2 = 5-pin pinheader
PCB # 120489-2

New from Parallax

The Parallax 36-position Quadrature Encoder Set conveniently provides rotational feedback for robot wheels.   This set was designed specifically for the Motor Mount and Wheel Kit (#27971), which is included with the Eddie and MadeUSA robotic platforms. Or, use this kit with your own custom robots or mechanical systems with ½" axles.
Key Features:

- Provides two out-of-phase outputs from within a single sensor assembly
- 36-position encoder disks, which resolve to 144 positions with the quadrature sensor output, are incised to grip 1/2" diameter axles
- Low power consumption doesn't drain your mobile power sources
- Dual-channel outputs provide both speed and directional information
- 6-pin single-row header accommodates a 4-wire or 6-wire interface accommodates

On www.Parallax.com search "29321." Price: $29.99.
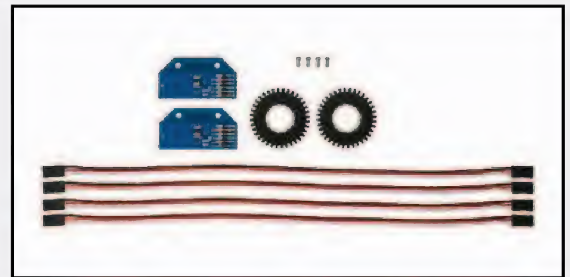
The Si1143 Proximity Sensor is great for non-contact gesture recognition in microcontroller applications. By measuring infrared light levels from the three onboard IR LEDs, gestures in the up, down, left, right and center select directions can be detected.
Key Features:

- Measures visible and IR ambient light levels, providing a range of operation from darkness to full sunlight
- Easy communication interface is compatible with any microcontroller
- Standard 0.1" header pins provide a convenient connection to breadboard or through-hole projects

On www.Parallax.com search "28046." Price: $29.99.

www.parallax.com   (120727-I)

New AIR module family for ZigBee® standard and new BoosterPack kit

Anaren, Inc. announced a new family of four Anaren Integrated Radio (AIR™) modules designed specifically to help OEMs develop products that wirelessly communicate in compliance with the ZigBee® standard. Based on the Texas Instruments Incorporated (TI) CC2530 low-power RF SoC (which operates using TI's Z-Stack™ firmware), the new family of AIR modules is bundled with *AIR Support for ZigBee* — a total solution that includes time-saving AIR-ZNP firmware (including 30+ code examples), pre-certification to applicable global, regulatory standards, and development tools like the Company's new BoosterPack for TI MSP430™ and Stellaris® LaunchPad development kits. A member of the ZigBee Alliance, Anaren unveiled the beta of its AIR module for ZigBee at the 2012 Consumer Electronics Show (CES).

Features and benefits of Anaren's new AIR module family for ZigBee standard applications (part number A2530x24xxx) include the following.

In concert with the launch of its new AIR module family for ZigBee standard applications, Anaren has also introduced a new BoosterPack featuring its new family of modules. The new CC2530 BoosterPack Kit (part number A2530E24A-LPZ) helps OEM engineers develop wireless applications using a TI LaunchPad for MSP430 or Stellaris MCUs.

www.anaren.com   (120727-IV)
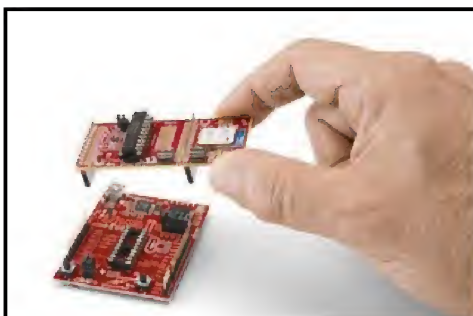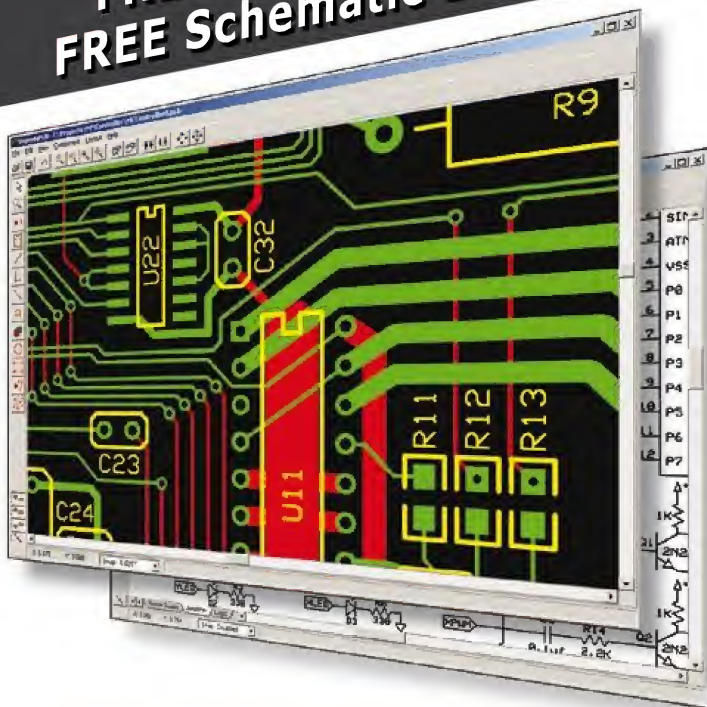
Cypress Gen4X devices double the industry's best noise immunity and lower total system cost

Cypress Semiconductor Corp. newest members of the TrueTouch® Gen4 touchscreen controller line are claimed to offer the industry's highest noise immunity, and were designed for cost-sensitive platforms including models for the rapidly growing China smartphone market. They deliver unparalleled performance in the presence of all noise sources — the biggest challenge faced by touchscreen designs — and support both in-cell and SLIM® single-layer structures to meet system-level price points needed to drive high volumes in emerging markets.

Improvements to the family's analog front-end minimized size and enabled twice the charger noise immunity of previous Gen4 offerings, which have gained popularity with industry leaders around the world due to outstanding performance in noisy environments.

Gen4X's immunity to charger noise is up to 15 V_{pp} from 1–500 kHz, with a 0.5-mm cover lens and a 22-mm-wide finger. Gen4X takes signal-to-noise ratio to unmatched heights, thanks to Cypress' unique ability to deliver on-chip 10 V Tx and proprietary Tx-Boost® multi-phase Tx solution.

Gen4X also delivers high-performance, multitouch functionality with customizable advanced features, including Cypress' industry-leading waterproofing capability. The entire Gen4 family works flawlessly with moisture on the touchscreen and/or with wet fingers. These features are enabled by Cypress' unique, patent-pending ability to provide both self and mutual capacitance sensing on the same chip.

Gen4X employs the same 32-bit ARM® Cortex® core that is at the heart of the Gen4 family to deliver the best combination of speed and low-power consumption. The family easily delivers a refresh rate of 150 Hz, and a deep-sleep mode that only draws 4.5 µW with wake-up initiated via address match on the COM port. Gen4X devices offer up to 36 capacitive sensing I/Os for ideal sensor pitch with screens up to 5 inches, support standalone CapSense® touch-sensing buttons, and drive both in-cell structures and SLIM single-layer sensors. Gen4X is the first touchscreen controller family to bring Superphone performance to emerging markets.

Samples of the new Gen4X controllers are now available to lead customers. Cypress expects to deliver volume production to lead customers in the first quarter of 2013. The family will be available in a 44-pin 5 mm x 5 mm QFN with 28 sense I/Os, a 44-pin 5 mm x 5 mm QFN with 33 sense I/Os, and a 48-pin 6 mm x 6 mm QFN package with 36 sense I/Os.

www.touch.cypress.com   (120727-III)

Rugged portable
RF/IF high-bandwidth signal recorder

Pentek, Inc.'s new Talon® RF/IF signal recording and playback system Model RTR 2727 is a rugged portable recorder suitable for military and aerospace applications. The system features recording and playback of IF signals up to 700 MHz with signal bandwidths to 200 MHz. It can be configured with 500 MHz 12-bit A/Ds or 400 MHz 14-bit A/Ds and an 800 MHz 16-bit D/A. Pentek's SystemFlow software allows turnkey operation through a graphical user interface (GUI), while the SystemFlow application programming interface (API) allows easy integration of the recording software into custom applications. Features of the new instrument include:

- Samples signals up to 500 MHz with built-in digital down-conversion
- Up to two channels of recording with SSD storage to NTFS RAID
- SystemFlow GUI control panel for fast, intuitive operation
- Remote system control and custom integration using SystemFlow API

At the heart of the recorder are the Pentek Cobalt® Series Virtex-6 software radio boards featuring A/D and D/A converters, DDCs (digital downconverters), DUCs (digital upconverters), and FPGA IP. This architecture allows the system engineer to take full advantage of the latest technology in a turnkey solution. Optional GPS time and position stamping captures this critical signal information within the recording.

The RTR 2727 has a portable, lightweight chassis with up to eight hot- swap solid state drives (SSDs), front panel USB ports and I/O connections on the side panel. Its extremely rugged, 100% aluminum alloy case is reinforced with shock absorbing rubber corners and an impact-resistant protective screen. Shock- and vibration-resistant solid-state drives (SSD) with combined capacity to 3.8 TB make the RTR 2727 a reliable, portable field instrument.
Available I/O includes audio and VGA video, RS-232/422/485 serial, multiple USB 2.0 and USB 3.0, eSATA, and dual GbE connections.
By using hot swappable SSDs, the recorders exhibit high immunity to shock and vibration for ground vehicles, ships and aircraft. The drive array capacity can be as large as 3.8 TB and supports RAID levels 0, 1, 5, or 6.
The built-in Windows 7 Professional workstation with an Intel Core i7 processor gives the user total flexibility in routing data to various drives, networks and I/O channels. Also, the user can install post- processing and analysis tools on the system itself to operate on the recorded data. The RTR 2727 is fully supported with Pentek's SystemFlow software for system control and turnkey operation. The software provides a GUI with point-and-click configuration management and can store custom configurations for single-click setup. The software also includes a virtual oscilloscope and spectrum analyzer to monitor signals before, during and after data collection.

www.pentek.com   (120727-V)

Open Source Hardware

Sébastien Bourdeauducq and Milkymist.

By
Tessel Renzenbrink
(Elektor TTF Editor)

Open source software has proved that it's worth its salt. Firmly rooted in two of humanity's more profound qualities, curiosity and collaboration, open source is driven by a need to access, share, modify and improve the source code. Because open source allows people to build on each other's solutions, it prevents duplication, cuts development costs, lowers entry barriers, and generally makes programming a lot more fun. With over half of the total number of mobile phones running on a Linux-based operating system it's safe to say that open source software has decisively established its place in the world.

For all of its success it isn't surprising to see the open source mindset ported to the hardware community and witness the emergence of Open Source Hardware (OSHW).
OSHW is hardware of which the design is made publicly available. The design may be shared, modified and (evidently) used to build hardware.

To make sharing and production more streamlined the OSHW community is working on standardization. They're developing collaboration tools and design software. And inspired by open source software they're publishing licenses which govern the rights and the restrictions of the user.

Operating at the outer reaches of OSHW is Sébastien Bourdeauducq, founder of an open source project called Milkymist [1]. He also organized several OSHW conferences, including the upcoming Exceptionally Hard & Soft Meeting taking place in Berlin December 28-30, 2012.

In a Skype chat Sébastien told me about opening up chip design and how the Milkymist code ended up in lower Earth orbit.

Open Source

Tessel: How did you get involved in Open Source Hardware?

Sébastien: I've always loved to understand how things worked. I started doing electronics and programming when I was a kid. My first contact with actual 'open source' was in 1999 when I switched to Linux — but I had touched other people's code before, which was published in magazines, books and BBSs [Bulletin Board Systems] even though it wasn't labeled 'open source' or 'free software'. I really liked Linux, after being frustrated with the restrictions of Windows that prevented you from touching the core of the operating system.

My first project that could be called 'open source hardware' was in 2002 — it was an infrared communication system for the TI89 calculator, with GNU Free Documentation License, a very simple thing. Then I went into wireless driver writing for FreeBSD (an open source UNIX-like operating system), which was more a mix of hardware and software, since I had to understand how the undocumented wireless hardware worked. But my biggest OSHW project today is Milkymist, which I started in 2007 (see **inset**). I wanted to open up chip design, something that few open source projects tackled and that's still the case today.

Innovate at the next level

Tessel: Why did you want to open up chip design?

Sébastien: One reason is philosophical — almost everyone's so called 'open source' electronics today is built on black box chips that they don't understand, and they don't try to. I found this barrier to understanding very frustrating.

Another reason is security: you can actually have hardware backdoors. (A backdoor is a

---

Milkymist (previously in Elektor)

The Milkymist One is a stand-alone video synthesizer used for rendering live video effects at music performances. Both on the hardware and software level the Milkymist is open source where possible. Remarkably, Milkymist has an FPGA-based System-on-Chip (SoC) that got custom developed and comes with a free Hardware Description Language (HDL) source code complement. Most components of the SoC, except the LatticeMico32 CPU core, are placed under the GNU General Public License.

The SoC programming and functionalities are described in some detail in Elektor USA, June 2012 [6].

The video rendering software Flickernoise is heavily inspired by Milkdrop, an open source plugin for the music player Winamp that generates visualizations. As open source projects go, many people have contributed to Milkymist [1].

Milkymist One hardware.

---

manipulation of a chip's firmware or hardware enabling an unauthorized party to gain access and disable or reprogram the chip. Backdoors can be installed on a chip during the production process. They're considered a genuine threat, especially in military hardware.)

Finally, the last reason is being able to innovate at the next level. This is actually quite successful, but outside the common OSHW community. Milkymist code has found its way into major scientific projects like the International Space Station (ISS) and particle accelerators.

In 2009 NASA's Jet Propulsion Laboratory (JPL) used the code of Milkymist SoC memory controller for its Space Communications and Navigation (SCaN) Program [2]. On October 15, 2012 Sébastien received a letter saying:

Dear Sébastien:
I'd like to take this opportunity to thank you for your open source work which helped our devel-

opment of the JPL Software Defined Radio that is part of the SCaN Testbed, which has now been installed on the International Space Station. It has been undergoing initial checkout and commissioning for a few months now, including the use of the dynamic ram controller that Greg Taylor built starting with your work. I think it's safe to say that many millions of bits have passed through the code you wrote in space by now.
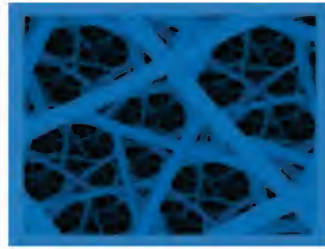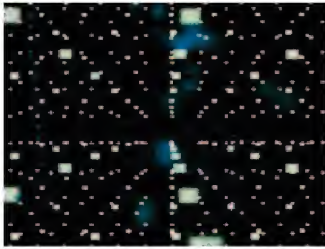
Thanks again,
James P. Lux, Co-Principal Investigator, SCaN Testbed

Tessel: You also contribute to open hardware initiatives of CERN, the European Organization of Nuclear Research. Many research institutes are secretive about their work, why does CERN embrace the OSHW approach?
Sébastien: Well, there are many different personalities in research institutes. It's true that many are quite secretive — sometimes on purpose, sometimes because they don't care and then academic publishers put research behind pay walls to make money. I think the high energy physics community is a little more open than the others, maybe because they do fundamental research that has no immediate commercial or military applications.
The CERN open hardware initiative is also the work of a handful of employees who believed in OSHW and wanted to make their institution part of it. They have created an online Open Hardware Repository where electronics engineers can collaborate on open hardware designs used in experimental physics. [3]. And in 2011 they published the CERN Open Hardware License. [4].

Exceptionally Hard & Soft Meeting
Tessel: Why are you organizing the Exceptionally Hard & Soft Meeting (EHSM)?
Sébastien: Many hacker and OSHW confer-

ences, including the three I have co-organized before EHSM [5], lacked in some areas of content. Most of the hardware technology is extremely simple and carefully goes round expensive and difficult areas. Which is kind of frustrating for the same reasons I started the Milkymist project. EHSM wants to go pretty far, for example our keynote speaker is a 17 year old boy who built a nuclear fusion reactor. EHSM is also trying to involve scientific institutions and research. We already have CERN. We'll probably have people from a quantum physics lab and perhaps even a hands-on experimental quantum physics workshop. We're also talking to another institute in Berlin which does very advanced materials research. Another change we want to make with regard to other conferences is that they are like 99% male.

Tessel: OSHW is different from open source software in that it will not only have to deal with copyright but also with patents. What are your thoughts on patents, do they harm OSHW?
Sébastien: Patents are so poorly implemented that they harm everything, including OSHW. One of the original ideas of the patent system was to encourage inventors to publish by giving them legal protection before the patent becomes public domain. Which is, in a way, an open source system because you make inventions public domain in the end, after the patent expires.
But with the patent system in place today, needless to say it failed miserably. It has just become some corrupt corporate battle ground. In particular when things that are not innovative at all get patented (see for example Apple's renowned patent on a rectangle with rounded corners). And only large corporations infringing on each other's' patents have an advantage there — they agree not to sue each other. But newcomers are easy targets; as a consequence the patent system stifles innovation rather than promote it.

The future of OSHW

Tessel: If the world would go open source, what would be the benefits?

Sébastien: I would see it as a huge learning opportunity to start with. But economically, I don't know. It's not very clear to me whether open innovations and the fact that you can collaborate far more easily with more people outweigh the benefits you may get from keeping your design details secret. But I like experimenting, and that's also why I got into projects like Milkymist. The positive thing I can say so far is the OSHW approach has been successful in attracting talented people.

Tessel: What role will OSHW play in the future? Will it replace proprietary closed hardware?

Sébastien: I don't see it happening in the near future. The reality is that most hardware is manufactured by companies like Apple or Intel who do not care the slightest bit about OSHW and hold a lot of know-how that they are not ready to release. And most OSHW people -at least those who don't go to EHSM- do not care about creating industry-level hardware. Of course, hopefully it'll get better in the long-term but it will be hard and will need a lot of mindsets to change.

(120643)

Internet References

[1] http://milkymist.org/

[2] http://spaceflightsystems.grc.nasa.gov/SOPO/SCO/SCaNTestbed/

[3] www.ohwr.org/

[4] www.ohwr.org/projects/cernohl/wiki

[5] http://ehsm.eu/

[6] www.elektor.com/110447

Philips PR9103
Portable Stroboscope (1956)
No disco — no flash in the pan

By Jan Buiting,
Managing Editor

The stroboscope was originally an optical apparatus in the scientific domain, designed to analyze object movement or displacement not normally perceivable with the human eye. In wikispeak, the object is 'frozen' in its movement or so it appears. By about 1940 mechanical engineers started to employ electrical versions for use in combination with their precision lathes, milling and other tooling instruments. Guess what, a few years on car repairmen found out that the stroboscope was also great for ignition timing adjustment using markers on the engine flywheel. All of these were relatively low-power stroboscopes, meaning the intensity of the flash was moderate and usually just enough to illuminate the object to be observed.

But then the disco hype struck in the early 1980s, and very high power stroboscope systems were installed in discotheques and clubs to produce relentless flash rhythms and light bursts on dance floors and stages. The crowds at the time loved the effect as it seemed to 'freeze' dancers, singers and band players in their movement, creating the illusion that people had moved several feet in a blink of an eye. Sadly the effect also proved hallucinating or hypnotizing to a degree, and outright hazardous to persons suffering from photosensitive epilepsy.

The instrument described here belongs in the hands of the Quality Inspection Officer of a 1950s mechanical workshop. It is totally unsuitable to disco use due to its rather weak flash level. It does however have a modern-day application the Philips designers could never have surmised like 60 years ago — more about this further on. Basically if you want to make a fast moving object like a turning fan blade appear to stand still so you can have a good look at it, you aim the parabolic reflector unit at the object, and turn the frequency control on the main unit until

optical interference and eventually zero-beat occurs. Philips in their PR9103 manual illustrate a 1950s TV deflection coil winding machine being examined for correct operation (**Figure 1**). With the stroboscope set to the right flash frequency, wire placement errors can actually be observed by an inspection engineer, instead of him seeing a big blur. FYI, TV deflection coil winding is a very fast and mechanically highly demanding process. If something goes wrong, the mess on the work floor is just unbelievable.

Mode d'emploi, s'il vous plait
The **inset** explains why this PR9103 came with a French user manual. Even with my limited knowledge of the language, I was able to detect from a few lines on page 1 that the manual was a dry translation from the equally dry Dutch. Philips, as you know was the pinnacle of Dutch electronics design and production for over 50 years, and in good Dutch fashion they were totally convinced of their language skills. Despite my best efforts I have been unable to find an English manual for the instrument. The manual is extensive in technical respects, explaining in detail how to deal with cases of multiple images appearing instead of one.

On to the circuit diagram (**Figure 2**). An ECC40

1

2

(6SN7) double triode rules, acting as a relaxation oscillator covering 15 Hz to 240 Hz in two ranges selected on the front panel. These frequencies equal 900 to 14,400 revolutions per minute (rpm) of a rotating object. External triggering is available to cover 0-60 Hz (EXT.I) and 60-240 Hz (EXT.II). For reasons I cannot fathom, the 0-15 Hz range is not available internally.

A type NSP2 (CV2296) 'strobotron' flash tube is used, to my great joy its tech data is on the web [1].

In practical use
Before actually trying to use the instrument I opened it up (**Figures 3** and **4**) and was greeted by an immaculate interior. The case with its large ventilation areas is familiar from the extensive series of professional electronics test instruments

3

4
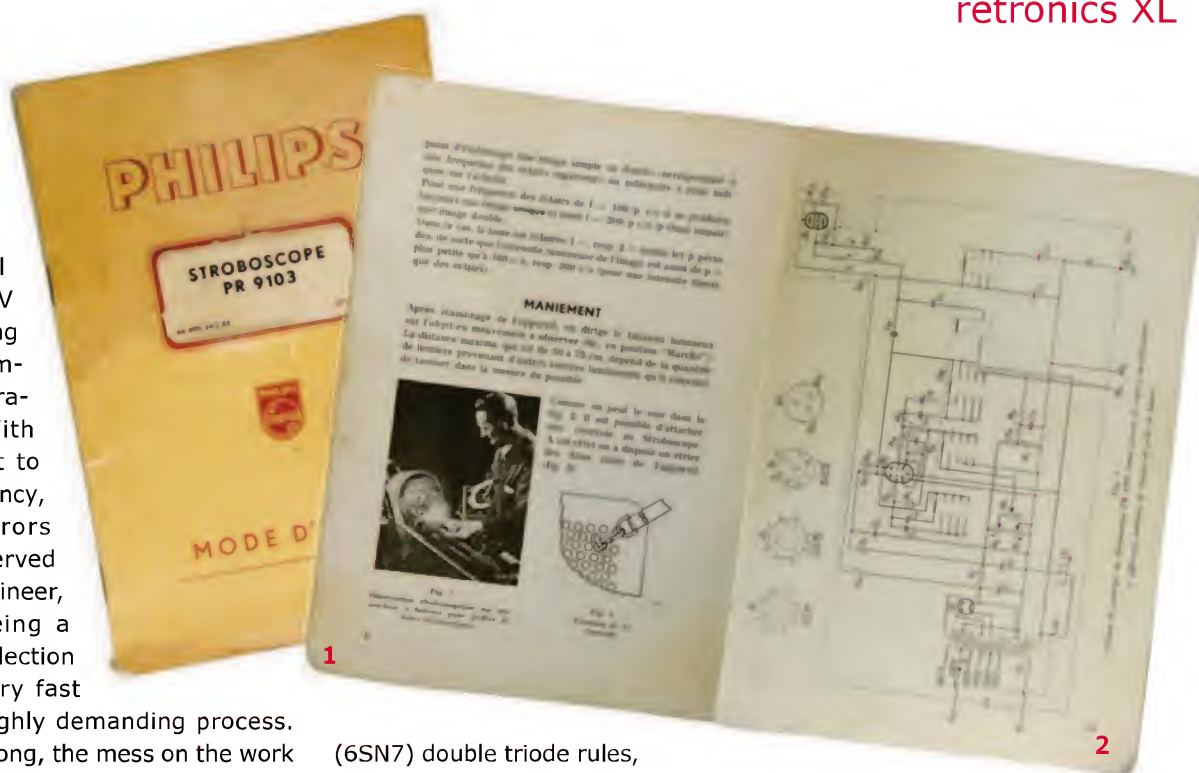
24 January 2013:
Free Retronics Webinar
Join the 'Best of Retronics'
webinar presented by Jan Buiting
and brought to you by element14
and Elektor. Attendance: free.
Seating: limited to "server
overload". Date/time: January
24, 2013, 4 pm CET. Sign up at:
www.elektor.com/webinars

built by Philips in the 1950s. My stroboscope was a rare case though of the leather carrying handle being intact after a good 60 years. The quality of the grey and black lacquer on the casing and front panel respectively is simply incredible. As you can see in **Figure 5** the shielded cable connecting the reflector unit to the control unit is in urgent need of repairs. Note the use of a Philips proprietary 3-pin polarized connector for high voltages.

As usual with old tube equipment, *Sleeping Beauty* has to be woken up gently using an adjustable power transformer. Starting from zero volts and gradually upping 20 volts every 10 minutes or so I found that the strobotron started to flash at about 70% of the nominal AC line voltage for the instrument. The flashing was accompanied by a distinctive ticking noise of the gas discharges in the tube. As it turned out I was at 20 Hz or so. I confidently raised the line voltage to the nominal level and was happy to see that my stroboscope was in NRN condition (no repairs needed). Colleagues started to gather round, marveling at the range of colors seen in the flash tube but also a bit disappointed at the intensity. I think I heard someone mumble 'John Travolta'? Time to switch off and read some more French.

Good vibrations

The instrument has a disarmingly simple method of calibration. Inside is a small solenoid operated at 6.3 VAC that causes a strip of thin metal secured at one side to vibrate at two times the 60 Hz powerline frequency (**Figure 6**). The extreme of the strip is painted red, bent upwards a little and visible through a slot cut in the large frequency dial (**Figure 7**). In the switch position 'Calibrate' you turn the frequency dial to 120 Hz, and point the flash light at the red indicator. Next you adjust the 'Freq. Cal. Min.' control on the front until the red indicator appears to be stationary instead of vibrating. Next, you turn the frequency dial to 240 Hz and adjust the 'Freq. Cal. Max.' control until a double, stationary image is obtained. A little slow movement is allowable. The Min. (60 Hz) and Max. (120 Hz) controls interact slightly and their settings have to be repeated a few times until both are satisfactory. The upshot is that the PR9103 instrument after calibration achieves <1 % overall frequency accuracy, independent of AC line voltage fluctuations of up to 10%.

Ah, la douce France

The stroboscope was given to me by an elderly lady who by incredible coincidence lives less than a mile from Elektor House in the quiet town of Limbricht, Holland. The story was: together with her husband and son, she would frequently travel to France in the 1970s and 80s to 'raid' antiques shows, rural places and car boot sales in France to buy French *bric-a-brac*, café interiors, and battered terrace chairs for selling on in Holland. The crux is: to most people in Holland, the French language is a nightmare from distant Highschool days, but at a more mature age the wine, weather and food are *fin bec, chouette, bon ton, exquisite,* and so on. By total contrast, this 3-person family living a good 150 miles closer to France than most flatland Hollanders was as Francophile as it gets, to the point of driving French saloon cars right through France in their search for valuables to haul Northwards, and sell on to their less refined countrymen. A brilliant business concept it was, for over 20 years.

As you may surmise the business dried up with the rise of the Internet, E-Bay, the general decline in interest in antiques and period furniture (from any country, for that matter), and the exploding cost of fuel. The fun over and most of the trade goods gone, the family decided to clear out their attic, backyard and garage where objects were discovered that years ago got thrown into a furniture deal somehow, but never sold. Like this Philips "lamp thing" no one wanted. Except me.
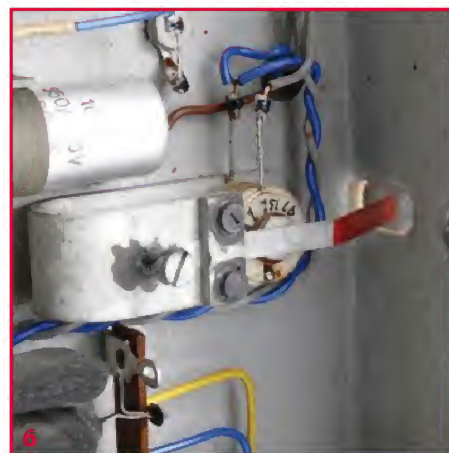
Into the smartphone & camcorder age

Apart from adjusting the ignition on a friend's 1965 Fiat 500 car I could not see much use for the stroboscope. My colleague Thijs Beckers however came up with the brilliant idea of checking out frame refresh rates (fps; frames per second) of video cameras as used in smartphones and camcorders. We pointed the cameras of a few Samsung and HTC smartphones at the stroboscope light, watched the images on the camera LCDs and slowly turned the frequency control. As you approach the fps specification, a black bar starts to roll slowly across the screen, just as in the old days when your B/W TV lost sync. If the bar is stationary (no matter inside or outside the image), you can read the camera's fps value from the stroboscope dial (**Figure 8**). Excellent eye/hand coordination training! We measured a few instances of 30 Hz fps on smartphone cameras and about 60 Hz (fps) on a camcorder. As a bonus, we were able to observe the mesmer-

izing effects of the flash tube igniting and passing through various colors before reaching its maximum intensity and then quenching again. The lamp off time is roughly the reciprocal of the flash frequency in Hz, i.e. 41 ms and 16 ms at 24 Hz and 60 Hz respectively. Some interesting cases of optical interferences were also observed under fluorescent lighting and under halogen lighting.

(120593)

[1] Jeremy M. Marmer's Virtual Tube Museum:
     www.tubecollector.org/cv2296.htm

Retronics is a monthly section covering vintage electronics including legendary Elektor designs. Contributions, suggestions and requests are welcome; please telegraph editor@elektor.com
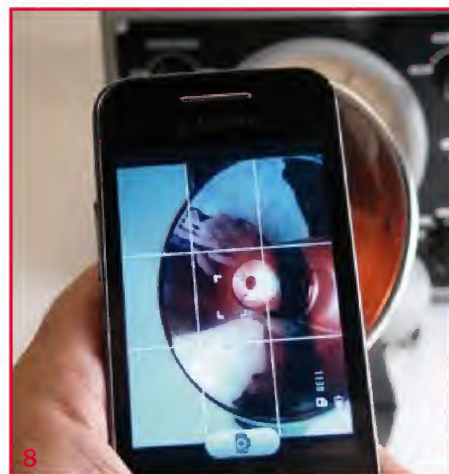
Hexadoku   Puzzle with an electronic touch

Welcome to the first Hexadoku installment of the year 2013. A new year, a fresh start.
If you've not participated so far, now's a good time to do so. Enter the right numbers or letters
A-F in the open boxes, find the solution in the gray boxes, send it to us and you automatically
enter the prize draw for one of four Elektor Shop vouchers.

The Hexadoku puzzle employs numbers in the hexadecimal range 0 through F. In the diagram composed of 16 × 16 boxes, enter numbers such that **all** hexadecimal numbers 0 through F (that's 0-9 and A-F) occur once only in each row, once in each column and in each of the 4×4 boxes (marked by the thicker black lines). A number of clues are given in the puzzle and these determine the start situation.

Correct entries received enter a prize draw. All you need to do is send us **the numbers in the gray boxes**.

Solve Hexadoku and win!

Correct solutions received from the entire Elektor readership automatically enter a prize draw for one Elektor Shop voucher worth $140.00 and three Elektor Shop Vouchers worth $60.00 each, which should encourage all Elektor readers to participate.

Participate!

Before March 1, 2013,

send your solution (the numbers in the gray boxes) to

www.elektor.com/hexadoku

Prize winners

The solution of the November 2012 Hexadoku is: **BD18A.**   The Elektor $140.00 voucher has been awarded to Ron Ware uit Witley (United Kingdom). The Elektor $60.00 vouchers have been awarded to Panagiotis S. Krokidis (Greece), Didier Pelegri (France), and P. Scheepers (Netherlands).

Congratulations everyone!

F	A		4							9		E	6
9			E	6		1			0		3	F			C
		B		3		4			C		F		7	
0	7				B	F			5	6				2	A
	C	F		2	4					D	8		E	0
			9	5		D			7			C	2	
	2	5	3		A		E	9		F			C	4	D
				C	B	A	2				
				9	7	E	F				
	E	6	C		2		3	0		7		D	1	A
		5	4		B			D		2	7	
	F	8		A	5				1	4		9	3
5	1				C	3		A	0				B	E
	7		9		A			E		1		6
D			F	1		E			3		B	4			0
E	3		0							A		7	1
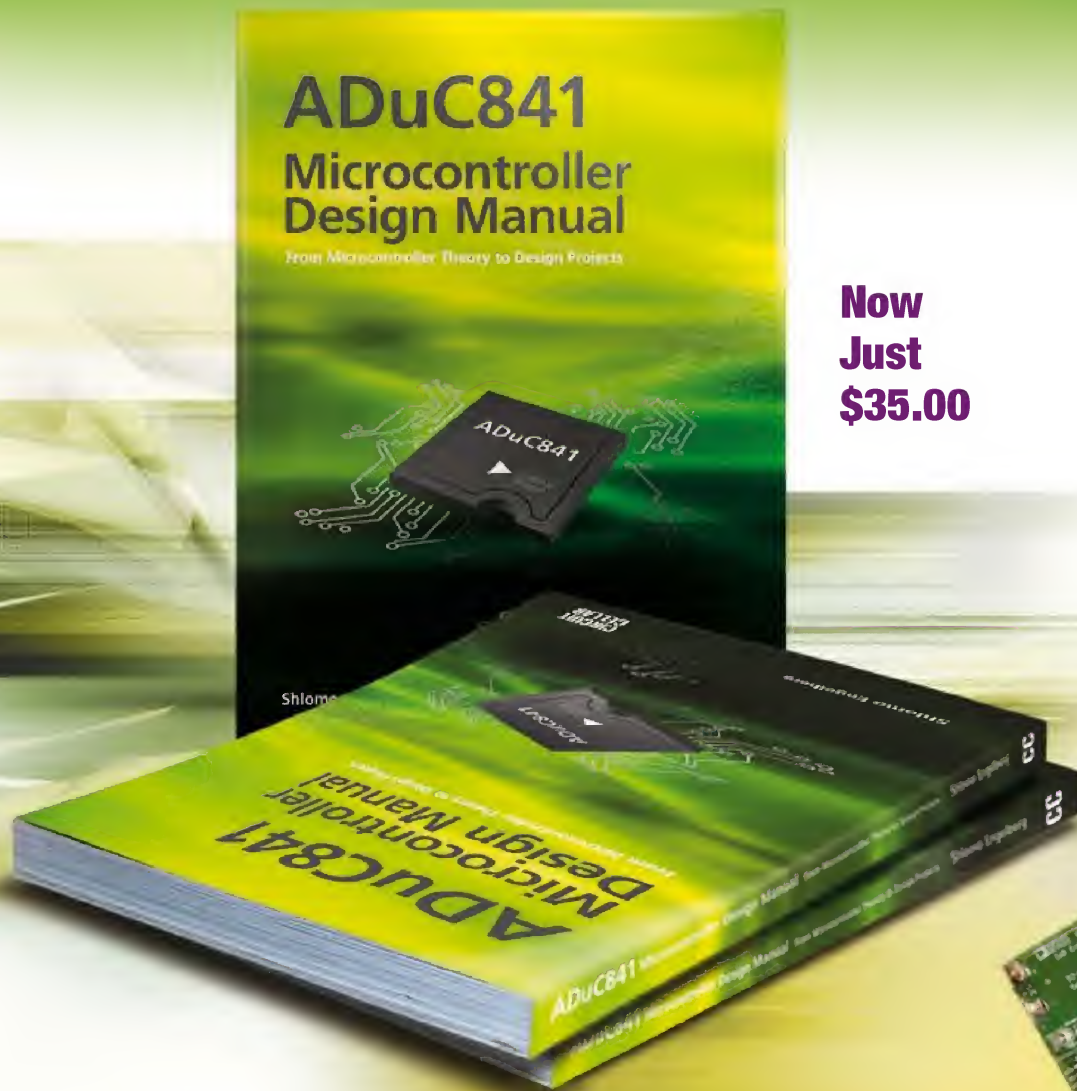
0	E	3	A	5	6	F	9	D	1	4	B	2	7	8	C
1	D	2	9	A	0	B	7	C	F	E	8	5	3	4	6
7	B	F	4	C	1	2	8	9	3	5	6	A	D	E	0
5	8	6	C	D	E	3	4	2	0	7	A	9	B	F	1
D	9	C	3	6	2	E	1	7	8	F	4	B	A	0	5
B	F	E	0	3	4	5	D	6	9	A	2	7	1	C	8
4	5	7	1	F	C	8	A	E	B	D	0	3	9	6	2
2	6	A	8	7	9	0	B	1	5	3	C	E	4	D	F
6	7	1	B	8	A	D	5	0	2	9	E	F	C	3	4
E	A	4	D	9	B	1	F	8	6	C	3	0	5	2	7
3	0	8	2	E	7	4	C	F	A	1	5	D	6	9	B
F	C	9	5	0	3	6	2	4	7	B	D	1	8	A	E
A	2	0	F	B	D	7	6	3	4	8	1	C	E	5	9
9	1	5	E	4	8	C	0	A	D	2	7	6	F	B	3
8	3	B	6	1	F	A	E	5	C	0	9	4	2	7	D
C	4	D	7	2	5	9	3	B	E	6	F	8	0	1	A

The competition is not open to employees of Elektor International Media, its business partners and/or associated publishing houses.

NEW

Limited Time Offer for GREEN and GOLD Members!
15% DISCOUNT
www.elektor.com/january

Programming step-by-step
1 Android Apps

This book is an introduction to programming apps for Android devices. The operation of the Android system is explained in a step by step way, aiming to show how personal applications can be programmed. A wide variety of applications is presented based on a solid number of hands-on examples, covering anything from simple math programs, reading sensors and GPS data, right up to programming for advanced Internet applications. Besides writing applications in the Java programming language, this book also explains how apps can be programmed using Javascript or PHP scripts. When it comes to personalizing your smartphone you should not feel limited to off the shelf applications because creating your own apps and programming Android devices is easier than you think!
244 pages • ISBN 978-1-907920-15-8
$56.40

More than 75,000 components
2 CD Elektor's Components Database 7

This CD-ROM gives you easy access to design data for over 11,100 ICs, 37,000 transistors, FETs, thyristors and triacs, 25,100 diodes and 2,000 optocouplers. The program package consists of eight databanks covering ICs, transistors, diodes and optocouplers. A further eleven applications cover the calculation of, for example, zener diode series resistors, voltage regulators, voltage dividers and AMV's. A colour band decoder is included for determining resistor and inductor values. All databank applications are fully interactive, allowing the user to add, edit and complete component data.
ISBN 978-90-5381-298-3 • $40.20

3
Free mikroC compiler CD-ROM included
Controller Area Network Projects

The aim of the book is to teach you the basic principles of CAN networks and in addition the development of microcontroller based projects using the CAN bus. You will learn how to design microcontroller based CAN bus nodes, build a CAN bus, develop high-level programs, and then exchange data in real-time over the bus. You will also learn how to build microcontroller hardware and interface it to LEDs, LCDs, and A/D converters.
260 pages • ISBN 978-1-907920-04-2
$47.60

Avoid interference and earth loops
4 USB Isolator

If your USB device ever suffers from noise caused by an earth loop or if you want to protect your PC against external voltages then you need a USB isolator. The circuit described in Elektor's October 2012 edition offers an optimal electrical isolation of both the data lines as well as the supply lines between the PC and the USB device.
Populated and tested board (incl. case)
Art.# 120291-91 • $101.40

LabWorX 2
5 Mastering Surface Mount Technology

This book takes you on a crash course in techniques, tips and know-how to successfully introduce surface mount technology in your workflow. Even if you are on a budget you too can jumpstart your designs with advanced fine pitch parts. Besides explaining methodology and equipment, attention is given to SMT parts technologies and soldering methods. Many practical tips and tricks are disclosed that bring surface mount technology into everyone's reach without breaking the bank. A comprehensive kit of
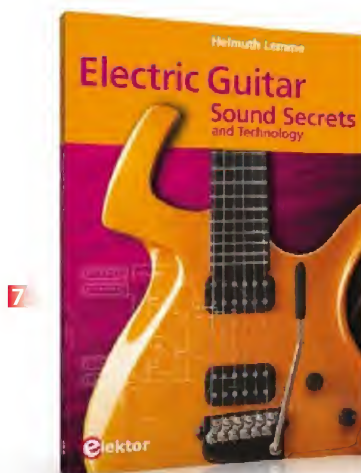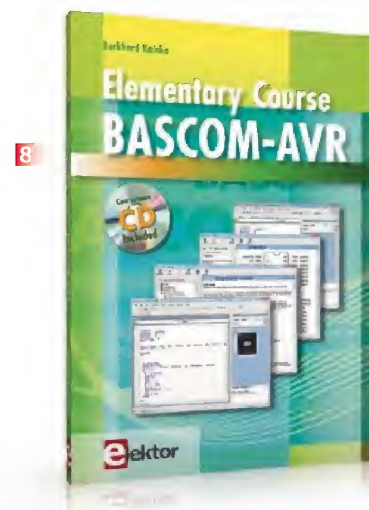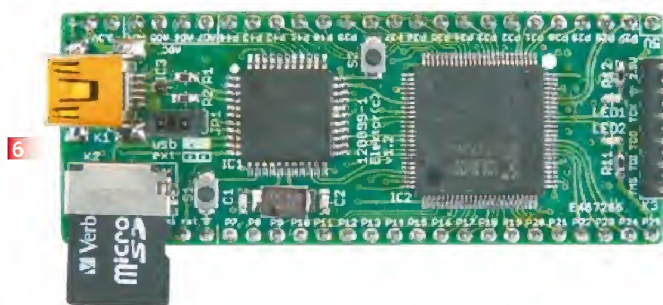
parts comprising all SMT components, circuit boards and solder stencils is available for readers wishing to replicate three projects described in this book.
282 pages • ISBN 978-1-907920-12-7
$47.60

Taming the Beast
6 FPGA Development Board

FPGAs are unquestionably among the most versatile but complex components in modern-day electronics. An FPGA contains a maze of gates and other circuit elements that can be used to put together your own digital circuit on a chip. This FPGA development board (designed in the Elektor Labs) shows how easy it is for any electronics enthusiast, whether professional or amateur, to work with these programmable logic devices.
Module, ready build and tested
Art.# 120099-91
See www.elektor.com/fpgaboard

Sound Secrets and Technology
7 Electric Guitar

What would today's rock and pop music be without electric lead and bass guitars? These instruments have been setting the tone for more than forty years. Their underlying sound is determined largely by their electrical components. But, how do they actually work? This book answers many questions simply, in an easily-understandable manner. For the interested musician (and others), this book unveils, in a simple and well-grounded way, what have, until now, been regarded as manufacturer secrets. The examination explores deep within the guitar, including pickups and electrical environment, so that guitar electronics are no longer considered highly secret. With a few deft interventions, many instruments can be rendered more versatile and made to sound a lot better – in the most cost-effective manner.
287 pages • ISBN 978-1-907920-13-4
$47.60

Free Software CD-ROM included
8 Elementary Course BASCOM-AVR

The Atmel AVR family of microcontrollers are extremely versatile and widely used. In Elektor magazine we have already published many interesting applications employing an ATmega or ATtiny microcontroller.

The majority of these projects perform a particular function. In this book we focus more on the software aspects. Using lots of practical examples we show how, using BASCOM, you can quickly get your own design ideas up and running in silicon.
224 pages • ISBN 978-1-907920-11-0
$56.40

Meet BOB
9 FT232R USB/ Serial Bridge/BOB

You'll be surprised first and foremost by the size of this USB/serial converter – no larger than the moulded plug on a USB cable! And you're also bound to appreciate that fact that it's practical, quick to implement, reusable, and multi-platform – and yet for all that, not too expensive! Maybe you don't think much of the various commercially-available FT232R-based modules. Too expensive, too bulky, badly designed, That's why this project got designed in the form of a breakout board (BOB).
PCB, assembled and tested
Art.# 110553-91 • $20.90

The world's first book with NFC technology integrated inside

10 Catch the Sun

The oldest known contactless connectivity technology dates back 2000 years to the Han dynasty in China. In that era, the Kongming lantern was invented: a small hot air balloon used primarily for transmitting military signals. The Kongming balloons have today been replaced by chips. Near Field Communication, or NFC, provides wireless connectivity over short distances based on semiconductor technology. This book links both technologies together.

Catch the Sun is the world's first book with NFC semiconductor technology integrated inside, while the content of this high-tech book is about the beautiful magic of low-tech ballooning. The book has multiple NFC chips inside that allow the book to connect to the internet, simply by touching an NFC-hotspot in the book with your NFC-enabled smartphone or tablet.

128 pages • ISBN 978-9-07545-861-9
$57.50

Associated 60-piece Starter Kit available

11 Fun with LEDs

This booklet presents more than twenty exciting projects covering LEDs, aimed at young & old. From an Air Writer, a Party Light, Running Lights, a LED Fader right up to a Christmas Tree. Use this book to replicate various projects and then put them into practice. To give you a head start each project is supported by a brief explanation, schematics and photos. In addition, the free support page on the Elektor website has a few inspiring video links available that elaborate on the projects. A couple of projects employ the popular Arduino microcontroller board that's graced by a galaxy of open source applications. The optional 60-piece Starter Kit available with this book is a great way to get circuits built up and tested on a breadboard, i.e. without soldering.

96 pages • ISBN 978-1-907920-05-9
$38.00

Circuits, ideas, tips and tricks

12 CD 1001 Circuits

This CD-ROM contains more than 1000 circuits, ideas, tips and tricks from the Summer Circuits issues 2001-2010 of Elektor, supplemented with various other small projects, including all circuit diagrams, descriptions, component lists and full-sized layouts. The articles are grouped alphabetically in nine different sections: audio & video, computer & microcontroller, hobby & modelling, home & garden, high frequency, power supply, robotics, test & measurement and of course a section miscellaneous for everything that didn't fit in one of the other sections.

ISBN 978-1-907920-06-6 • $55.70

Package Deal: 12% off

13 AVR Software Defined Radio

This package consists of the three boards associated with the AVR Software Defined Radio articles series in Elektor, which is built around practical experiments. The first board, which includes an ATtiny2313, a 20 MHz oscillator and an R-2R DAC, will be used to make a signal generator.

The second board will fish signals out of the ether. It contains all the hardware needed to make a digital software defined radio (SDR), with an RS-232 interface, an LCD panel, and a 20 MHz VCXO (voltage-controlled crystal oscillator), which can be locked to a reference signal. The third board provides an active ferrite antenna.

Signal Generator + Universal Receiver +
Active Antenna: PCBs and all components +
USB-FT232R breakout-board
Art.# 100182-72 • $133.00

110 issues, more than 2,100 articles
14 DVD Elektor 1990 through 1999

This DVD-ROM contains the full range of 1990-1999 volumes (all 110 issues) of Elektor Electronics magazine (PDF). The more than 2,100 separate articles have been classified chronologically by their dates of publication (month/year), but are also listed alphabetically by topic. A comprehensive index enables you to search the entire DVD. What's more, this DVD also contains the entire 'The Elektor Datasheet Collection 1...5' CD-ROM series.
ISBN 978-0-905705-76-7 ● $111.30

Enhanced second edition
15 Design your own Embedded Linux Control Centre on a PC

The main system described in this book reuses an old PC, a wireless mains outlet with three switches and one controller, and a USB webcam. All this is linked together by Linux. This book will serve up the basics of setting up a Linux environment – including a software development environment – so it can be used as a control centre. The book will also guide you through the necessary setup

and configuration of a webserver. New edition enhancements include details of extending the capabilities of your control center with ports for a mobile phone (for SMS messaging) and the Elektor "thermo snake" for lowcost networked real-time thermal monitoring of your house and outbuildings. Now you can additionally also send all kinds of useful temperature and sensor warnings to a mobile phone. All software needed will be available at the Elektor website.
416 pages ● ISBN 978-1-907920-02-8
$55.70

140 Minutes video presentation and more
16 DVD Feedback in Audio Amplifiers

In this Masterclass we address several aspects of feedback in audio amplifiers. The focus of this Masterclass, although not entirely math-free, is on providing insight and understanding of the issues involved. Presenter Jan Didden provides a clear overview of the benefits that can be obtained by feedback and its sibling, error correction; but also of its limitations and disadvantages. Recommended to audio designers and serious audio hobbyists!
ISBN 978-907920-16-5 ● $40.20

Counter for alpha, beta and gamma radiation
17 Improved Radiation Meter

This device can be used with different sensors to measure gamma and alpha radiation. It is particularly suitable for long-term measurements and for examining weakly radioactive samples. The photodiode has a smaller sensitive area than a Geiger-Müller tube and so has a lower background count rate, which in turn means that the radiation from a small sample is easier to detect against the background. A further advantage of a semiconductor sensor is that is offers the possibility of measuring the energy of each particle.
Kit of parts incl. display and programmed controller
Art.# 110538-71 ● $57.30

.Magazine

Intelligent LCR meter

In 1997 Elektor published a top class, cutting edge LCR meter with technical specs matching professional equipment of that time. At last, in the March 2013 edition we're able to present a new LCR meter Totally up-to-date technology wise, it also surpasses its famous predecessor in terms of specifications and capabilities. This device can be used stand alone, or in combination with a PC via a USB connection.

Raspberry Pi Prototyping Board

The Raspberry Pi is a very affordable, credit card size, computer development system ready for connecting lots of peripherals such as a keyboard and a display. It's ideal as a a base for many (homebrew) applications. To simplify the connection of your own electronic circuits to the Pi a prototyping board was designed comprising a stabilized power supply and a prototyping area with solder pads. The prototyping board is simply plugged on top of the Raspberry Pi.

Thermo Book

Here's an uncommon thermometer? For this design we've not only extended the functionality, but also opted for an original case. Shaped as a book the display of this measurement circuit shows either temperature or humidity, where switching occurs automatically after a fixed time, or 'manually' by clapping your hands in the room.

Article titles and magazine contents subject to change; please check www.elektor-magazine.com
Elektor USA March 2013 edition published February 20, 2013.

See what's brewing @ Elektor Labs 24/7

Check out
www.elektor-labs.com
and join, share, participate!

ORDERING INFORMATION

To order contact customer service:

Phone: 860-875-2199
Fax: 860-871-0411
Mail: Elektor US
111 Founders Plaza, Suite 300
East Hartford, CT 06108
USA
E-mail: order@elektor.com
On-line at www.elektor.com/store

Customer service hours: 8:30 AM-4:30 PM EST Monday-Friday. Voice mail available at other times. When leaving a message please be sure to leave a daytime telephone number where we can return your call.

PLEASE NOTE: *While we strive to provide the best possible information in this issue, pricing and availability are subject to change without notice. To find out about current pricing and stock, please call or email customer service.*

COMPONENTS

Components for projects appearing in Elektor are usually available from certain advertisers in the magazine. If difficulties in obtaining components are suspected, a source will normally be identified in the article. Please note, however, that the source(s) given is (are) not exclusive.

PAYMENT

Orders must be prepaid. We accept checks or money orders (in US $ drawn on a US bank only), VISA, Mastercard, Discover, and American Express credit cards. We do not accept C.O.D. orders. We also accept wire transfers. Add $20 to cover fees charged for these transfers.

TERMS OF BUSINESS

Shipping Note: All orders will be shipped from Europe. Please allow 3–4 weeks for delivery. Shipping and handling via airmail: $20.00 per order.

Returns
Damaged or miss-shipped goods may be returned for replacement or refund. All returns must have an RA #. Call or email customer service to receive an RA# before returning the merchandise and be sure to put the RA# on the outside of the package. Please save shipping materials for possible carrier inspection. Requests for RA# must be received 30 days from invoice.

Patents
Patent protection may exist with respect to circuits, devices, components, and items described in our books and magazines. Elektor accepts no responsibility or liability for failing to identify such patent or other protection.

Copyright
All drawing, photographs, articles, printed circuit boards, programmed integrated circuits, diskettes, and software carriers published in our books and magazines (other than in third-party advertisements) are copyrighted and may not be reproduced (or stored in any sort of retrieval system) without written permission from Elektor. Notwithstanding, printed circuit boards may be produced for private and personal use without prior permission.

Limitation of liability
Elektor shall not be liable in contract, tort, or otherwise, for any loss or damage suffered by the purchaser whatsoever or howsoever arising out of, or in connection with, the supply of goods or services by Elektor other than to supply goods as described or, at the option of Elektor, to refund the purchaser any money paid with respect to the goods.

MEMBERSHIPS (US & CANADA ONLY)

Order memberships on-line at www.elektor.com/members

All memberships begin with the current issue. Expect 3–4 weeks for receipt of the first issue.
Membership renewals and change of address should be sent to:

Elektor US
P.O. Box 462228
Escondido, CA 92046

E-mail: elektor@pcspublink.com

Memberships may be paid for by check or money order (in US $ drawn on a US bank only). We accept Mastercard, VISA, Discover and American Express credit cards.

For gift memberships, please include gift recipient's name and address as well as your own, with remittance. A gift card will be sent on request. Memberships may be cancelled at any time for a refund of all unmailed issues.

Does your membership expire soon?

Renew it on-line at www.elektor.com/members

NEW

DEEP MEMORY
MIXED SIGNAL OSCILLOSCOPES
PORTABILITY & PERFORMANCE

PicoScope	3204 MSO	3205 MSO	3206 MSO
Channels	2 Analog 16 Digital		
Bandwidth	60 MHz	100 MHz	200 MHz
Buffer memory	**8 MS**	**32 MS**	**128 MS**
Resolution (enhanced)	8 bits (12 bits)		
Signal generator	Function generator + AWG		
Price	$1070	$1400	$1730

www.picotech.com/PCO496   CALL TOLL FREE: 1 800 591 2796